Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 08, No. 01, March 2022.

DES22: DES-BASED ALGORITHM WITH IMPROVED SECURITY

Malek M. Barhoush[1], Bilal H. Abed-Alguni[2], Rafat Hammad[3], Mohammad Al-Fawa'reh[1] and Rana N. Hassan[4]

(Received: 8-Oct.-2021, Revised: 19-Dec.-2021, Accepted: 3-Jan.-2022)

ABSTRACT

We live in a world where the Internet has become the backbone of most of our dealings. The Internet has turned this big planet into a small village. The Internet can be reached by everyone, everywhere, at any time. Some authors predict that the number of various types of devices capable of connecting via the Internet will reach 75.44 billion by 2025. These devices vary from low-processing power processors to heavy-processing power processors. It often requires the protection of mobile data between devices. These devices that have limited energy and resources require the protection technology to be adapted. The time it takes to encrypt a message using Data Encryption Standard (DES) is much less than the time it takes to encrypt the same message using Advanced Encryption Standard (AES). The problem with DES is that the key size is small and this makes it vulnerable to brute force attack. This paper gives complete guidelines for adapting the original DES and making it more secure, along with improving its performance compared to the existing standard encryption algorithms, such as AES. The proposed approach improves the original DES security by extending the key size of DES without affecting the cost of DES. The new algorithm is called DES22 and is convenient for low-processing power devices, such as wireless sensors. DES22 has three variants for key size: 128 bits, 256 bits and 512 bits. The paper also proposes another improvement to DES through random permutation and the distribution of the initial permutation and final permutation tables between the encryption and decryption algorithms. The experimental results show that DES22 is more secure and faster than AES.

KEYWORDS

Data encryption standard, DES, Advanced encryption standard, AES, IoT.

1. INTRODUCTION

The Internet has revolutionized the world of communication, where the transmission of various data of large volumes is carried out at a high speed and in less time, making people in contact with each other at any time and from anywhere. This revolution has transformed our planet into a small village. The Internet has transformed society, especially with the presence of Internet of Things (IoT) technologies and cloud computing platforms that connect everything, everywhere at any time. This of course has a positive impact on the progress of mankind, as transactions have become electronic thus turning our life more flexible. On the other hand, the development of technologies and the Internet has facilitated many cybercrimes. The Internet is available to all its users, all over the world, which makes it vulnerable to attack and therefore, it needs protection [1]-[3].

According to [4]-[5], by 2025, about 75 billion devices will be able to communicate with each other *via* the public network. Examples of these devices are: laptops, desktops, IoT, wireless sensors, smartphones, tablets, cars, airplanes, trains, biomedical machines, switches, routers and so on. Some of these devices have processors with high capabilities, while some other devices have processors with medium capabilities and others have processors with very modest capabilities.

Data collected from different types of devices connected to the Internet is a source of intelligent analysis through which decisions are made to improve a particular goal [5]. This data needs to be transferred quickly and needs to be protected from various types of attack.

Encryption is one of the traditional and important means to obtain confidentiality over the public

1. M. M. Barhoush (ORCID: 0000-0002-1146-7293) and M. Al-Fawa'reh (ORCID: 0000-0002-5621-4126) are with Department of Information Technology, Yarmouk University, Irbid, Jordan. Emails: malek@yu.edu.jo and fawareh@yu.edu.jo
2. B. H. Abed-Alguni (ORCID:0000-0002-7481-4854) is with Department of Computer Science, Yarmouk University, Irbid, Jordan. Email: bilal.h@yu.edu.jo
3. Rafat Hammad (ORCID: 0000-0001-9698-7345) is with Department of Information Systems, Yarmouk University, Irbid, Jordan. Email: rafat.hammad@yu.edu.jo
4. R. N. Hassan is with Ministry of Education, Irbid, Jordan. Email: rananaimh77@gmail.com

Internet. It also provides other services, such as authentication, integrity and non-repudiation [2], [6]. Since the Internet allows multiple processes to connect with no physical direct connections, their information may flow among intermediate eavesdropper(s); therefore, it is important to protect their privacy. Various algorithms have been developed to provide encryption services, such as Data Encryption Standards (DESs), Blowfish, Twofish and Advanced Encryption Standards (AESs).

The emergence of cloud computing, mobile computing, big data and the IoT has necessitated the need to develop cryptographic algorithms that help improve their efficiency, speed and confidentiality and reduce battery consumption [7].

Underlying cryptography architecture comprises the following terms: Plain text, cipher text, encryption and decryption algorithms and shared key. The implementation of encryption and decryption algorithms is known to the public, but the secret key is not. In the encryption process, the plain text is converted into cipher text. Meanwhile, in the decryption process, the cipher text is back-converted into the plain text [8].

Encryption algorithms are classified into two main categories: symmetric and asymmetric. In the symmetric cryptography algorithm, the key used for the encryption process is the same as that used for the decryption process. The most important things that distinguish symmetric encryption from asymmetric encryption are the speed of execution and the amount of memory used to hold the key. In the case of symmetric encryption, the execution speed is higher and the amount of memory used for the key is less, compared to asymmetric encryption [9].

DES is one of the symmetric algorithms used for data security between 1977 and 2000, as declared by the National Institute of Standards and Technology [10]. The size of the key in the DES is 56 bits, which means that the process of trying all combinations of the key to decrypt a cipher text that was encrypted with DES is equal to 256 attempts. This is one of the main reasons that led to the cancellation of dealing with DES. In this paper, we propose new models for DES that work with larger key and same block size.

In this paper, the subsequent sections are organized as follows: the literature review is presented in Section 2. A description of DES is introduced in Section 3. In Section 4, the proposed design of the new algorithm DES22 is presented. DES22 security analysis is introduced in Section 5. In Section 6, DES22 performance evaluation and results' analysis are discussed. In Section 7, DES and AES complexity analysis is presented. Finally, the conclusion is introduced in Section 8.

2. LITERATURE REVIEW

Pfitzmannl and Anmann [11] outlined the construction of G-DES algorithm, where they proposed modifications to the DES algorithm and expected that these modifications would increase the speed of DES and make it the fastest among its peers. G-DES allows the user to enter a key with a length of 768 bits and this key is sub-divided into 16 sub-keys of 48 bits each, removing the sub-keys generation activity from the original DES algorithm. In addition, G-DES allows users to enter the substitution boxes created by the user, create initial permutation (IP) and its reverse (FP) or (IP^{-1}) and finally enter the expanding permutation table. These suggestions did not find practical reality, as no implementation of GDES was carried out.

Eli Biham [12] claimed that the speedup of executing his new DES proposal using Alpha-64 architecture compared to the original DES execution is 5. He provided a parallel form of DES implementation running on parallel SIMD model. The author claimed that DES needs 16000 instructions per block and with his new DES implementation, a block can be encrypted within the time it takes to execute 260 instructions. However, Eli Biham's proposal cannot work with all different modes of DES, in addition to that it does not enhance the DES security.

Anderson et al. [13]-[14] proposed an alternative form of the Advanced Encryption Standard (AES) algorithm, which they called Serpent algorithm. Serpent algorithm uses S-boxes as does DES, but the way they designed the S-Box made Serpent more efficient than AES. The size of the data block that the Serpent algorithm handles is 128 bits, the size of the key is 128, 192 or 256 bits and the number of rounds in Serpent algorithm is 32, each round working with four 32-bits in parallel. The data block is represented in little-endian format. Serpent algorithm generates 33 sub-keys of 128-bit length. Figure

1 shows the workflow of Serpent Encryption Algorithm. As you can see from Figure 1, Serpent algorithm starts with IP and ends with FP. Each round performs three activities: key mixing, substitution and linear transformation. The Serpent algorithm needs special hardware to work with and it does not find practical reality.

Alani proposed DES96 algorithm in order to enhance the security of DES [15]. The key size of DES96 algorithm is 84-bit. The sub-key is generated *via* S-boxes and XOR process. The IP table contains 84 entries, the IP sub-divides the 84-bits key into 3 parts: 48-bits, 28 bits and 8 bits. The first 48 bits are converted to 32 bits via S-boxes, the next 28 bits are permuted and converted to 16 bits and the last 8 bits are sub-divided into four adjacent 2 bits and then every 2 bits are XORed. As a result, 4 bits are produced. These 3 parts contribute to the creation of the sub-keys. Alani's proposal increases the security of DES a little bit.

In [16], the authors suggested a new technique to enhance the security of DES. They proposed that the content of the following tables or lists should not be static: IP table, the inverse permutation table, substitution boxes, expansion box and shrinking box. In other words, the transposition and substitution processes should not rely on static tables. The key input of their algorithm is tied with a random number, the value of which will determine the rotation value for the above-mentioned tables or lists. The idea is brilliant, but the way they translated their idea was not perfect. The number of possible IP along with corresponding inverse tables is 64, the number of possible expansion and corresponding shrinking tables is 32 and the number of possible S-boxes is 8*64. In this case, their suggested proposal has 64*32*8* 64 possible transposition and substation tables or lists. The brute force attack for this model is 64*32*8*64*2^{56} tries.

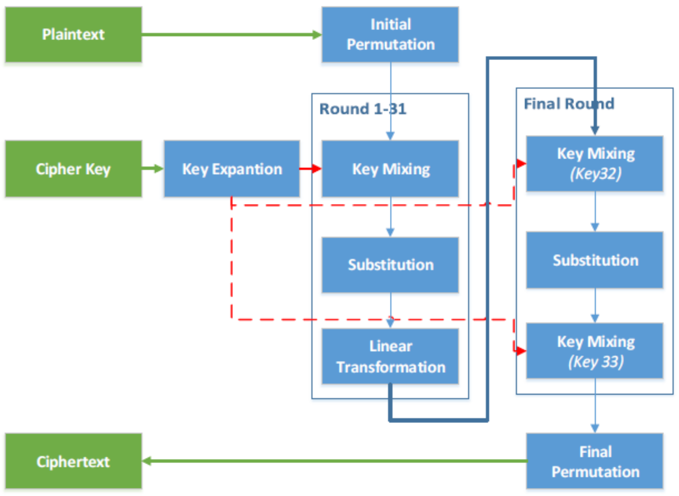

Figure 1. Workflow of serpent encryption algorithm [14].

In [17], the authors changed some components within DES. They replaced the eight S-boxes with one S-box and this change reduced the complexity of the DES hardware design circuit and thus reduced the execution time and energy consumed. Furthermore, the authors apply pipeline technology to 3DES encryption and decryption processes, which leads to reduce the latency of 3DES encryption and decryption processes.

In [18], the authors introduced a new cryptography algorithm based on the AES algorithm, with three rounds and a 128-bit key size. They replaced the AES S-box with their own 16-entries S-box. They claimed that their algorithm is secure. In the world of cryptography, the fewer rounds the algorithms have, the less confusion and diffusion the cipher has. Therefore, the security of [18] algorithm needs justification.

In [19], the authors suggested a semi-AES-128 algorithm, in which they reduced the number of rounds by a half compared to the original AES and as a result, the execution time of the new code was reduced by 35%. The security of their algorithm needs justification.

In [20], the authors proposed a pipelined implementation of AES that could work with multicore processors. Their proposal has 1.7 speedup compared with the original AES implementation. This type of algorithm needs multi-core processors to deliver better performance.

In [21], the authors proposed instruction set extensions to increase the performance and reduce the instructions count of AES implementation. They showed in their experiments that the speedup of their proposal is 10. The authors utilized embedded RISC processor, SPARC V8 architecture and superscalar processor for their test.

In [22], the authors designed a hardware for AES S-box and its inverse using 78 XOR and 36 AND gates, along with a coupled quadratic congruential generator (CQCG). The CQCG generates a random sequence for their design. This design increases the performance of AES algorithm.

In [23], the authors proposed a modified version of AES; they generated S-box table *via* PN Sequence Generator component and the same PIN is used to generate the input key. Their design increased the security of AES; however, it did not improve AES's performance. Table 1 summarizes the literature review section.

Table 1. Literature review summary.

Algorithm	**Description**	**Advantages**	**Disadvantages**
G-DES [11].	A variant of DES algorithm with a key length of 768 bits.	It seems to have more security.	There is no practical reality for the algorithm.
Parallel DES [12].	A parallel DES proposal using Alpha-64 architecture.	Its speedup is 5.	It cannot work with all different modes of DES. Key size is 56 bits.
Serpent algorithm [13]-[14].	The authors replace AES S-box with S-boxes used in DES; Serpent has 32 rounds. Serpent utilizes parallelism.	Serpent is more efficient than AES.	Serpent algorithm needs special hardware to work with.
DES96 algorithm [15].	DES96 has a key length of 84 bits.	The security of DES96 is better than that of DES.	DES96 security is not sufficient.
New DES [16].	The authors suggest that all DES tables should be rotated randomly.	The security of the algorithm is better than that of DES.	The security of this algorithm is not enough.
A variant of DES [17].	The authors replace the eight S-boxes of DES with one S-box	The complexity is reduced compared with DES.	Key size is 56 bits.
AES-based algorithm [18].	The modified AES has three rounds and a 128-bit key size.	The modified AES is faster than the original AES.	The cipher has less confusion and diffusion property.
simi-AES-128 [19].	The number of iterations is reduced to the half compared with AES.	The time latency is reduced to 35% compared with AES.	The security of the algorithm needs justification.
Pipelined- AES [20].	A variant implementation of AES utilizing pipeline.	Pipelined-AES has 1.7 speedup compared with the original AES.	The algorithm needs multi-core processors to deliver better performance.
Instruction set extensions [21]	Adding new instruction set to support AES performance.	The authors claim that the speed up is 10.	The proposal needs a change in the hardware architecture.
Hardware S-box [22].	Designing a hardware S-box using 78 XOR and 36 AND gates.	The design increases the performance of AES algorithm.	The design needs special hardware.
Modified version of AES [23].	The authors generated s-box table *via* PN Sequence Generator component.	The security of AES is increased.	AES performance is a problem.

3. DATA ENCRYPTION STANDARD (DES)

In this section, we will walk through the important points in the life cycle of DES to understand how to improve its security.

3.1 DES Workflow

DES deals either with block or stream cipher; the inputs of DES block encryption algorithm are blocks which a plain text length of 8 bytes or 64 bits and a key length of 56 bits. DES stream cipher works with a byte or many bytes. DES includes many simple operations, such as: substitution, permutation, word expansion, 6 to 4 bytes shrinking and XOR operation. In the block encryption process, the plain text is divided into blocks of 8 bytes long. For each input block, DES algorithm iterates 16 times before producing the 8 bytes cipher block. Each round requires 48-bits sub-key, so the algorithm needs 16 sub-keys of 48-bits length each. These sub-keys are generated out from a 56-bits key in a process called sub-keys generation. Figure 2, Figure 4 and Figure 5 show the DES encryption workflow.

The DES encryption starts with IP to the input block of 8 bytes length, where the bits of the input block are reordered according to the predefined IP table. Figure shows both IP and FP tables. As an example, bit numbers 58, 50, 42, 34, 26, 18, 10, 2 of the input block become the first eight bits of the output block, while bit numbers 60, 52, 44, 36, 28, 20, 12, 4 of the input block become the second eight bits of the output block, and so on.

The output of permutation process is a block of 2 words. It is divided into two halves: left word (L-word) and right word (R-word), on the basis that the word is equal to four bytes. The two words are passed through 16 rounds, where within each round, the same activities are repeated. These activities are summarized as follows: R-word expansion process, round key XORing process, expanding R-word, shirking process, 32-bit permutation process and L-word XORing process. Figure 2, Figure 4 and Figure 5 depict the 16 rounds for DES encryption process.

Figure 5 shows the black box for each round, where the L-word and R-word resulting from round i are used as an input to the next round. Figure 6 depicts the white box for each DES ith round. The R-word is passed through; expansion process, where the expansion process uses a predefined expansion table; this table describes how to expand the 32-bit R-word into 48 bits. The result is XORed with 48 sub-key associated with iteration ith number. Then the result is shrunk into 32 bits using predefined S-boxes. The resulting 32 bits are XORed with L-word. Finally, the L-word is swapped with the R-word, so that the current round L-word becomes the next round R-word and the current round R-word becomes L-word for the next round.

The DES encryption workflow is summarized as follows:

1- Divide the input plain text into blocks of length 8 bytes / 2 words, each word is 32 bits long.
2- If the last block is less than 8 bytes, then pad it with zeros.
3- Generate 16 sub-keys out from 56-bits input key, each sub-key has a length of 48 bits. Name each sub-key: SK (1), SK (2), …, SK (16).
4- While there is a block of plain text not processed yet, do steps 4 to 7. Otherwise, go to step 10.
5- Pass the current block into the permutation process called IP.
6- Name the permutated 2 words: L-word and R-word.
7- Pass the two words L-word and R-word through 16 rounds, where each round performs the same following operations with R-word:
    a. Pass R-word into the expansion process to produce 48 bits. See Figure 6.
    b. XOR the expanded 48 bits with the corresponding SK(i), where “i” is an integer number in the range from 1 to 16 and refers to round number. See Figure 6.
    c. Pass the result from the previous step through predetermined S-box to shrink it into 32 bits. See Figure 6.
    d. Then, pass the Shrunk 32 bits into the permutation process, then the result is XORed with L-word; the result is the new value of L-word. See Figure 5.
    e. Swap the new value of L-word with R-word, so that the current round is finished and the next round is configured with new values of L-word and R-word. See Figure 5. A summary of each round’s activity is shown in Figure .
8- After 16 rounds are finished, the final LW and RW are passed into the final permutation called IP-1 and then the current cipher block is ready.
9- Go to step 4.
10- Concatenate all cipher blocks and then the whole encryption process is done.

In the substitution operation, each text pattern is uniquely substituted by cipher pattern, while in the permutation operation, the bits are distributed in an organized and studied way utilizing a table that guides the distribution mechanism [10]. The theory behind substitution and permutation is to hide the properties of the plain text, thus making it difficult for the attacker to break the cipher without knowing the key [27].

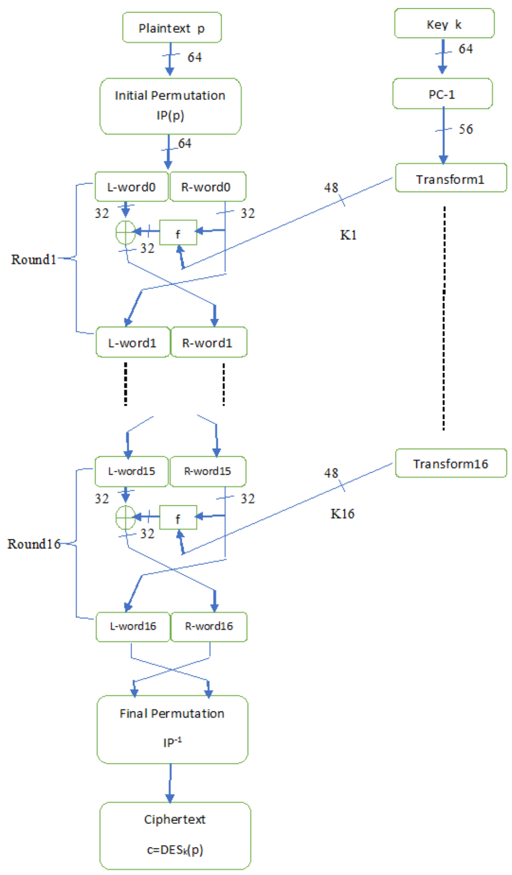

Figure 2. DES encryption process [24].

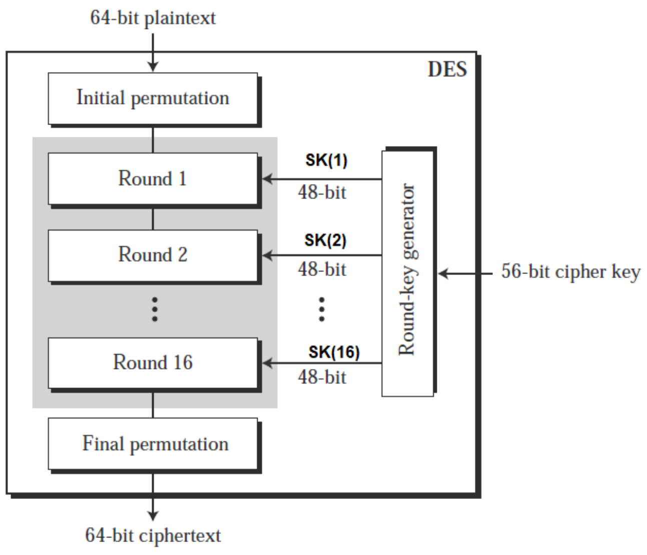

Figure 4. DES general structure [26].

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

Figure 3. IP and IP inverse tables [25].

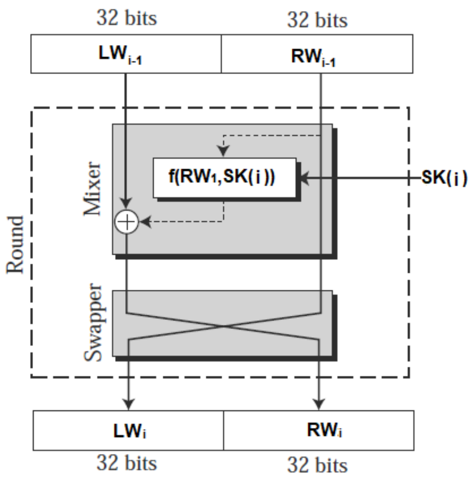

Figure 5. Black box for each round [26].

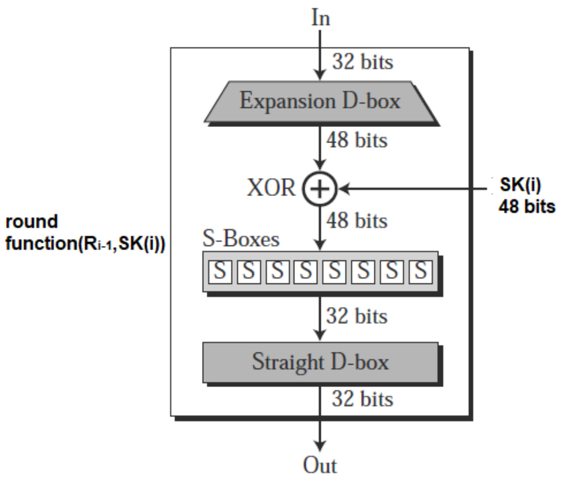

Figure 6. White box for each DES round [26].

In [10], the author states that for a stronger secure algorithm, it is recommended to have a large block size, a large key size, a large number of iterations producing staging product cipher, a strong key expansion and complex operations within each iteration.

3.2 Sub-keys Generation

The DES encryption and decryption processes use a key with a length of 56 bits. Originally, the entered key is 64 bits / 8 bytes long. The uppermost bit of each byte is discarded; therefore, the remaining bytes are 56 bits long. The DES encryption or decryption processes pass through 16 rounds, where each round uses a sub-key derived from a 56-bits key. A process called sub-keys generation derives 16 sub-keys out from a 56-input key. The process starts by permuting the 56-bits key using a table called PC-1, then the result is broken into two parts; left half (C) and right half (D), where each part has a length of 28 bits. The generator iterates 16 times doing the following activities to generate 16 sub-keys with a length of 48 bits:

a) Rotate left each of the two parts C and D either by one or two bits. For iteration number 1, 2, 9 and 16, the left rotation is one bit. Otherwise, the left rotations are two bits. As a result, the overall left rotation for each C and D is 28 bits [24]**.**
b) Pass a copy of C and D into another permutation process using a table called PC-2, which will generate a sub-key SK(i) with a length of 48 bits.

Figure 8 depicts the key scheduling process. Figure shows the two tables PC-1 and PC-2.

The DES decryption algorithm has the same workflow as the DES encryption algorithm, except that it uses the key scheduling order in reverse and this is why the authors describe this kind of algorithm as Feistel [8].

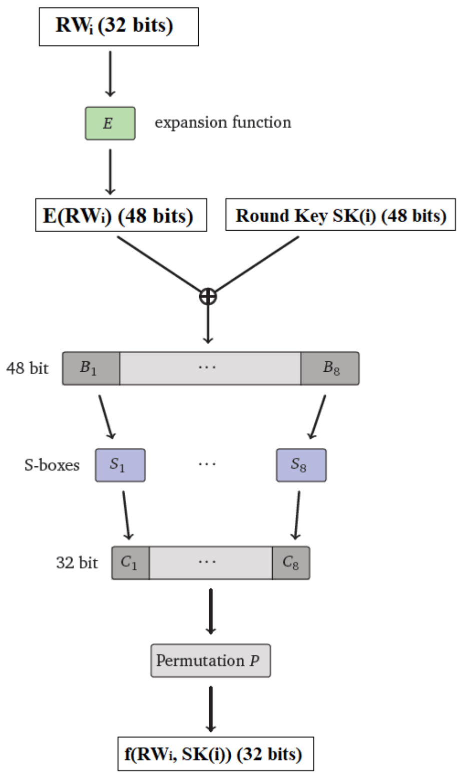

Figure 7. Summary of round activities.

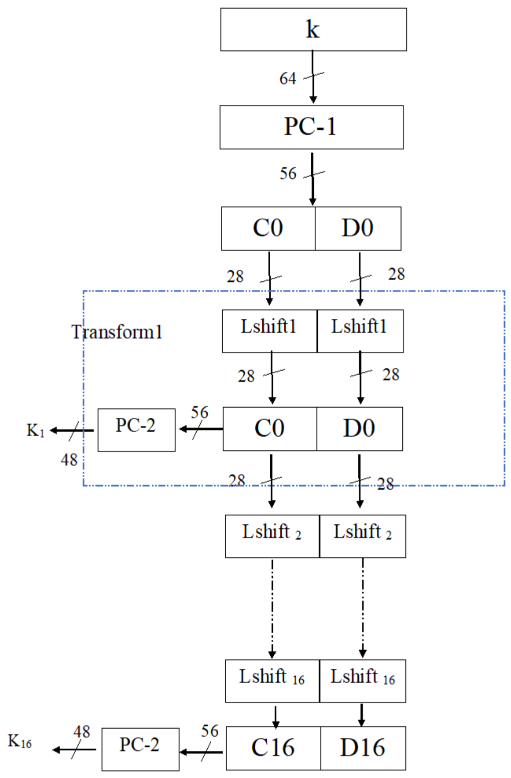

Figure 8. Key schedule for DES encryption [24].

4. The Proposed Design of DES22

We call the proposed design DES22, where the number 22 refers to the year 2022. DES22 transforms 64-bits plaintext blocks into 64-bits cipher text blocks. DES22 is a substitution transformation with 16 rounds, with the key sizes of 128, 256 or 512 bits. The IP table along with its inverse (IP^{-1}) are dynamically created before the start of the encryption process. Therefore, the parameters for the DES22 algorithm are plain text or cipher text, a key of length (128, 256 or 512) bits and a dynamic initial permutation table (DIP) along with its reverse (DIP^{-1}). The DES22 algorithm is an open-source algorithm, while the key, DIP and (DIP^{-1}) should be only be only shared with the sender and the receiver.

4.1 Dynamic IP and IP Inverse

The DES process starts with IP for each plain text block and terminates with its inverse (IP^{-1}) before producing the final value of cipher text block. The lookup table for IP and (IP^{-1}) are standard and fixed. These two processes do not add any security value to the DES algorithm and may hide a back door threat [24].

We provide a procedure that dynamically creates a table for IP process (DIP) and the corresponding inverse table for IP^{-1} process (DIP^{-1}). The terms DIP and DIP^{-1} can be used to denote processes or tables depending on the context. Since these two tables do not give any secrecy value to DES, we distribute them among the encryption and decryption algorithms. We use the DIP process at the end of the encryption process, while we use the DIP^{-1} at the beginning of the decryption process. The two tables DIP and DIP^{-1} are randomly generated; so, it is necessary that they are available only to the sender and the receiver. This distribution increases the security of DES. The following pseudo-code shows the dynamic creation of the DIP and its corresponding DIP^{-1} tables. Algorithm 1 describes the Dynamic creation of the tables IP and IP inverse.

Algorithm 1. Creating dynamic permutation tables DIP and DIP^{-1}

```
Input: no input
Output: return two lists DIP and $DIP^{-1}$
Steps:
    1-  Create empty lists l1, DIP, $DIP^{-1}$ each with empty 64 entries
    2-  Initiate the list l1 with integer values range from 1 to 64
    3-  count = 0, index = 0
    4-  while count < 64 do
    5-      index = choose a random integer exists in the list l1
    6-      DIP [count] = index
    7-      DIP-1[index] = count
    8-      Remove the value index from the list l1
    9-      count = count + 1
    10- end while
    11- return DIP and $DIP^{-1}$
    12- end
```

It should be noted that to have better randomness in producing DIP and DIP^{-1} tables from the previous pseudo-code, we need to deal with true random generation. It should be noted that the number of probabilities of the tables generated by the previous pseudo-code is equal to the factorial of 64 (64!); this huge number of probabilities increases the security of the DES algorithm.

4.2 Key of Size 128, 256 & 512 Bits

The original DES deals with a key of size 56 bits; this key is permuted using a lockup table called PC1, where table PC1 is shown in Figure . The proposed DES22 deals with a key-unit of 64-bits, therefore, PC−1 table is modified from 56 entries into 64 entries. We called the new table Modified PC−1 (MPC-1). Round key **permutation table PC−2 [24].** **key.** shows the permutation table of 64 input-key; the matrix MPC-1 does not exclude any bit of the input key-units. We believe that the original design of DES excludes 8 entries from the input key because of the need for parity bits in data transfer. In the proposed DES22, we do not need to exclude any bits from the input key. The proposed DES22 is designed to support key lengths of 128 bits, 256 bits or 512 bits. The input key is divided into multiples of 64 bits; we will call them multiple key-units. Therefore, 128-bit key is composed of 2 key-units, 256-bit key is composed of 4 key-units and 512-bit key is composed of 8 key-units. The process of creating sixteen 48-bits sub-keys starts with permuting each key-unit using the MPC−1 matrix.

If the key input is a 128 bits long key, 2 key-units, then each of the two key-units passes through eight rounds (n = 8) to generate sixteen sub-keys. Each key-unit is subdivided into its corresponding two 32-bit parts called C and D. For each key-unit, the round key generator iterates 8 times doing the following activities to generate 8 sub-keys with a length of 48 bits:

a) For the first iteration, rotate left each of the two parts C and D by three bits.

b) For the remaining iterations, rotate left each of the two parts C and D by four bits. As a result, the overall left rotations for each C and D are 31 bits.
c) Pass a copy of C and D into the permutation process using the table called PC-2, which is the same table used in the original DES algorithm. This permutation generates a sub-key SK(i) with a length of 48 bits, where i = 1, 2, 3……16.

$PC-1$							
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

$PC-2$							
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	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Figure 9. Initial key permutation table PC−1 and Round key permutation table PC−2 [24].

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

Figure 10. The permutation table of 64-input key.

If the key input is a 256 bits long key, 4 key-units, then each of the four key-units passes through four rounds (n = 4) to generate sixteen sub-keys. Each key-unit is subdivided into its corresponding two 32-bit parts called C and D. For each key-unit, the round key generator iterates 4 times to perform the following activities to generate 4 sub-keys with a length of 48 bits:

a) For the first iteration, rotate left each of the two parts C and D by seven bits.
b) For the remaining iterations, rotate left each of the two parts C and D by eight bits. As a result, the overall left rotations for each C and D are 31 bits**.**
c) Pass a copy of C and D into another permutation process using the table called PC-2. This permutation generates a sub-key SK(i) with a length of 48 bits, where i = 1, 2, 3……16.

If the key input is 512 bits long, 8 key-units, then each of the eight key-units passes through two rounds (n = 2) to generate sixteen sub-keys. Each key-unit is subdivided into its corresponding two 32-bit parts called C and D. For each key-unit, the round key generator iterates twice to perform the following activities to generate two sub-keys with a length of 48 bits:

a) For the first iteration, rotate left each of the two parts C and D by fifteen bits.
b) For the second iteration, rotate left each of the two parts C and D by sixteen bits. As a result, the overall left rotations for each C and D are 31 bits**.**
c) Pass a copy of C and D into another permutation process using the table called PC-2. This permutation generates a sub-key SK(i) with a length of 48 bits, where i = 1, 2, 3……16.

Figure 11 depicts the new key scheduling process, where **n** represents the number of iterations to generate 48-bits sub-keys, MPC-1 represents the modified permutation table PC-1, each C and D is with a length of 32 bits. The subscript i in the words key-unit represents the i^{th} key-unit according to the table below.

Table 2 shows the relation between the length of input key and the number of key-units, as well as the number of iterations to generate n 48-bits sub-keys and the number of overall 48-bits sub-keys.

The reason behind applying total left rotation for each key unit by 31 bits is that this process produces unique sub-keys out from each key-unit. Table 3 depicts the values between key-units and their corresponding sub-keys. Table 4 shows the results of encrypting a plain text using different key-units.

4.3 Ignoring Weak Keys

In [8], Forouzan mentioned that some keys are considered weak or semi-weak and should be taken care of. Weak keys are four: all bits that have a value of zero, all bits that have a value of 1 or all bits that have a value of zero in the C part and a value of 1 in the D part, or just the opposite. On the other hand, there are 12 semi-weak keys; there values in the hexadecimal system are: [01FE 01FE 01FE 01FE, FE01 FE01 FE01 FE01, 1FE0 1FE0 0EF1 0EF1, E01F E01F F10E F10E, 01E0 01E1 01F1 01F1, E001 E001 F101 F101, 1FFE 1FFE 0EFE 0EFE, FE1F FE1F FE0E FE0E, 011F 011F 010E 010E, 1F01 1F01 0E01 0E01, E0FE E0FE F1FE F1FE, FEE0 FEE0 FEF1 FEF1]. Forouzan also mentioned that there are keys called potential weak keys. The use of these keys leads to generate only four distinct sub-keys.

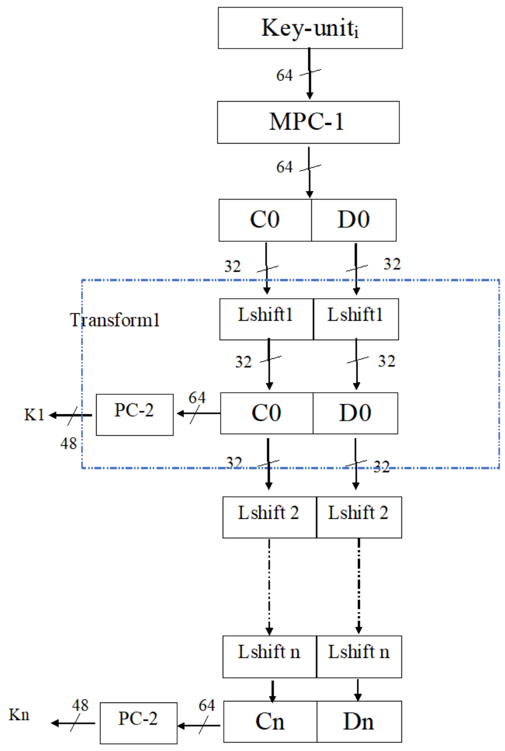

Figure 11. The new key scheduling process.

Table 2. The relation between the length of input key and the number of key-units.

Key length in bits	Number of key-units	Number of iterations to generate (n) 48-bits sub-keys **or** Number (n) of 48-bits sub-keys generated out of each key-unit.	Number of overall 48-bits sub-keys
128	2	8	16
256	4	4	16
512	8	2	16

Table 3. The generated sub-keys based on different key-units.

No. of Key-units	**Key-unit value**	**Sub-keys**

2	KU[1] = 0x123456789ABCDEF0	sub_key[1]= 0x439f4cc6ac1d
		sub_key[2]= 0xb1ed6a19d3a3
		sub_key[3]= 0x719f15ca29dc
		sub_key[4]= 0x37c4f6731623
		sub_key[5]= 0x3a833d047bdc
		sub_key[6]= 0x270afce38c2b
		sub_key[7]= 0x9a2f783d51f4
		sub_key[8]= 0x439f4cc6ac1d

	KU[2] = 0x22234512987ABB23	sub_key[9]= 0x1c9d64820556
		sub_key[10]= 0xadc724504649
		sub_key[11]= 0xb9b2718075a8
		sub_key[12]= 0x7273d4f24832
		sub_key[13]= 0x43617f953050
		sub_key[14]= 0x530bab10ac8e
		sub_key[15]= 0x950ece0b60e3
		sub_key[16]= 0x1c9d64820556

	KU[1] = 0x123456789ABCDEF0	sub_key[1]= 0xb1ed6a19d3a3
		sub_key[2]=0x37c4f6731623
		sub_key[3]=0x270afce38c2b
		sub_key[4]=0x439f4cc6ac1d

	KU[2] = 0x22234512987ABB23	sub_key[5]=0xadc724504649
		sub_key[6]=0x7273d4f24832
		sub_key[7]=0x530bab10ac8e
		sub_key[8]=0x1c9d64820556

4	KU[3] = 0xFEDCBA9876543210	sub_key[9]=0x86b6b30bd8b2
		sub_key[10]=0xe4495b1506ad
		sub_key[11]=0xd7d488ee0133
		sub_key[12]=0xe84cb54ebea8

	KU[4] = 0x9988776655443322	sub_key[13]=0x8d15e8a29c2b
		sub_key[14]=0xf97445c1a41c
		sub_key[15]=0x6372a78629d1
		sub_key[16]=0x8e0bc74c70cc

8	KU[1] = 0x123456789ABCDEF0	sub_key[1]=0x37c4f6731623
		sub_key[2]=0x439f4cc6ac1d

	KU[2] = 0x22234512987ABB23	sub_key[3]=0x7273d4f24832
		sub_key[4]=0x1c9d64820556

	KU[3] = 0xFEDCBA9876543210	sub_key[5]=0xe4495b1506ad
		sub_key[6]=0xe84cb54ebea8

	KU[4] = 0x9988776655443322	sub_key[7]=0xf97445c1a41c
		sub_key[8]=0x8e0bc74c70cc

	KU[5] = 0x123456789ABCDEF0	sub_key[9]=0x37c07c631633
		sub_key[10]=0x61af4dc6a81d

	KU[6] = 0x22234512987ABB23	sub_key[11]=0x7373d4a2c8a2
		sub_key[12]=0x1cbf64820172

	KU[7] = 0xFEDCBA9876543210	sub_key[13]=0xe4096b1526e5
		sub_key[14]=0xe948b55efe28

	KU[8] = 0x9988776655443322	sub_key[15]=0xf97415c1845c
		sub_key[16]=0x8e0fc35c30cc

What distinguishes DES22 is that when generating a key, it makes sure that none of the above-mentioned keys are used. DES22 makes sure that each key-unit is not equal to the four weak keys and does not equal any of the twelve semi-weak keys. In case that the key-unit is among the list of possible weak keys, this condition is confirmed by checking that all the 16 sub-keys must be distinct.

5. DES22 Security Analysis

DES22 is a security improvement to the DES algorithm. It uses three key sizes (128, 256 and 512 bits), which is a clear improvement in terms of security. Also, it uses dynamic permutation tables DIP and DIP^{-1}. Obviously, in order to break an encryption/decryption algorithm that uses n-bits key using brute force attack, this needs 2^n tries. On the other hand, using dynamic permutation tables of 64 entries, the attack process becomes more difficult, as it needs 64! attempts. Table 5 summarize the number of attempts to break DES22 *via* brute force attack. The time needed for a brute force attack is equal to the number of attempts multiplied by the speed of one attempt by the world's fastest supercomputer. The following formula expresses the time interval for a brute force attack.

Table 4. The results of encrypting a plain text using different key-units.

Key-units in hexadecimal value	Plain text in hexadecimal value	Cipher text in hexadecimal value

KU[1] = 0x123456789ABCDEF0
KU[2] = 0x22234512987ABB23
	0x9474b8e8c73bca7d	0x7f7fe23bf6189e77

KU[1] = 0x123456789ABCDEF0
KU[2] = 0x22234512987ABB23
KU[3] = 0xFEDCBA9876543210
KU[4] = 0x9988776655443322
	9474b8e8c73bca7d	8e4ea022e7964db0

KU[1] = 0x123456789ABCDEF0
KU[2] = 0x22234512987ABB23
KU[3] = 0xFEDCBA9876543210
KU[4] = 0x9988776655443322
KU[5] = 0x123456789ABCDEF0
KU[6] = 0x22234512987ABB23
KU[7] = 0xFEDCBA9876543210
KU[8] = 0x9988776655443322
	9474b8e8c73bca7d	67d8f66f41850f16

$\text{Time}_{\text{brute force attack}} = \text{No. of attempts} * \text{speed}_{\text{one attempt}}$.

Assume that the world’s fastest supercomputer has the speed of 415.5 peta FLOPS and assume that the speed of DES22 encryption process is 415.5 FLOPS, then each brute force attempt takes 415.5 FLOPS; in this case, this supercomputer can perform 10^15 encryption activities each second.

2^128 * 1/ peta flops = 340282366920938463463374.60743177 sec = 340282366920938463463374 years = 3.4 * 10^23 years

2^256 * 1/ 442,010 teraflops = 2.619671257150657121412886247114e+59= 8.3069230630094403900713034218482e+51 years

64! = 1.2688693218588416410343338933516e+89

Table 5. The relationship between DES22 key size and brute force attack.

Key size	No. of attempts using brute force attack
DES22 using 128-bits key	2^{128} attempts
DES22 using 256-bits key	2^{256} attempts
DES22 using 512-bits key	2^{512} attempts
DES22 using 128-bits key + dynamic permutation	(2^{128} + 64!) attempts
DES22 using 256-bits key + dynamic permutation	(2^{256} + 64!) attempts
DES22 using 512-bits key + dynamic permutation	(2^{512} + 64!) attempts

6. DES22 Performance Evaluation and Results' Analysis

The DES22 is implemented with three different key lengths, 128, 256 and 512 bits, in addition to using the dynamic permutation table as key extension. For performance evaluation, we ran three algorithms, such as: DES22, DES and AES, to compare the time executions among them. We use the terms data-units and cipher-units to describe a unit of data or cipher with a length of 64 bits. The table below summarizes the execution time for encrypting n data-units and decrypting n cipher-units using DES22, DES and AES implementations. For AES implementation, we run the one with a key of 128 bits. In this research, we get the best DES implementation from the site [28], as well as the best AES implementation from the site [29].

The test was performed on Intel laptop Intel® core™ I7-6500 CPU @ 2.50 GHz 2.60 Hz and 16 GB RAM, Windows 10 professional 64-bits operating system. Table 6 depicts the system we use.

Table 6. System specifications.

Processor	Intel® core™ I7-6500 CPU @ 2.50 GHz 2.60 Hz
RAM	16 GB
System type	64-bit operating system, x64-based processor

Table 7 shows the speed of encrypting a variable number of data-units (8 bytes / 64 bits) using DES, AES, DES22, Parallel DES, Serpent, DES96, New DES and a variant of DES. The abbreviation "enc" in the table header means encryption. DES22-128 means DES22 with 2 key-units (128 bits), 256 means 4 key-units (256 bits) and 512 means 8 key-units (512 bits). It is noticeable that DES22 is close to the speed of DES and almost three times faster than AES. It is also noted that DES22 is faster and more secure than all its peers. Parallel DES is faster -but less secure- than DES22 due to the size of the key.

Table 7. The execution time for encrypting data-units.

Time in milliseconds for enc [No. of data-units]										
No. of data-units	**DES**	**AES**	**DES22 128**	**DES22 256**	**DES22 512**	**Parallel DES [12].**	**Serpent [13] [14].**	**DES96 [15].**	**New DES [16].**	**A variant of DES [17].**
1	.01	.0708	.01	.012	.012	.004	.032	.015	.014	.025
2	.014	.07	.019	.02	.018	.004	.31	.022	.02	.035
1K	12	46	9	9	10	4	26	21	19	28.8
2K	24	90	22	23	21	8	52	35	33	55
4K	41	150	31	33	34	11	87	68	62	94
10K	93	325	79	79	90	19.7	180	144	132	241
20K	274	696	150	169	181	60	550	412	390	680
50K	526	1912	418	420	224	157	1024	811	795	2,782
100K	1339	3481	895	915	925	337	2695	2027	1912	2,678
1M	7887	27610	7113	7201	7386	1985	15,879	12012	11152	16,562

The following chart in Figure shows the execution time to encrypt 1 mega data-units (8 megabytes) using DES, AES, DES22, Parallel DES, Serpent, DES96, New DES and a variant of DES.. From time perspective, AES is 3 times slower than DES and DES22. Since the security of DES22 is far better than those of both AES and DES and DES22 is faster than AES, it is obvious that DES22 has a better performance and security for simple and complex processors.

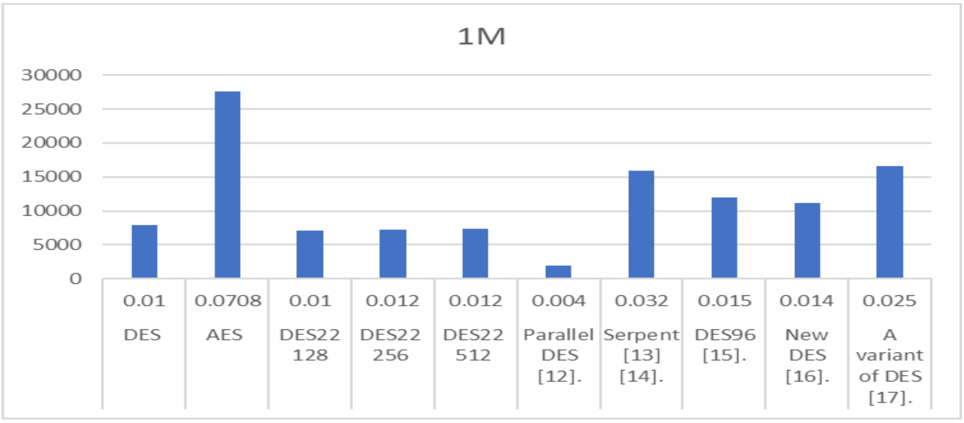

Figure 12. The execution time to encrypt or decrypt 1 Mega data-units.

7. DES and AES Complexity Analysis

DES and DES22 are Feistel cipher algorithms, which accept a block of 64-bit plain text and produces a block of 64-bit cipher text, or *vice versa*. The block is passed through an initial permutation and then through 16 stages of product ciphers, in which at each stage the following operations are performed on the block:

1. Splitting 64 bits into two halves.
2. Swapping the two halves.
3. Combining the two halves.
4. 32-bit to 48-bit expansion *via* P-box.
5. Using XOR.
6. Shrinking 48 bits into 32 bits *via* S-boxes.
7. 32-bit permutation.

The DES S-boxes were designed under the supervision of IBM researchers and they considered certain types of attacks in their design; one of these attacks is differential cryptanalysis. Recently, it is discovered is that DES is immune to differential attack due to the S-boxes design [30] and this of course is reflected in the DES22. DES is susceptible to linear and Davies-Murphy attacks [30]-[31] and this also is reflected in the DES22.

On the other side, AES is non-Feistel cipher and sometimes called Rijndael. Rijndael was selected among four other peers, MARS, RC6, Serpent and Twofish. The reason behind the superiority of AES lies in the following characteristics: security, cost, algorithm & implementation. AES is a block cipher, the block size is 128-bit and the key size varies: 128-bit, 192-bit or 256-bit.

In AES, the plain text is divided into 128-bit blocks, where each block is converted into a 4X4 array called state. The state is XORed with round key and then the result is passed through 10, 12 or 14 rounds, noting that the length of the key determines the number of rounds. In each round, the following operations are performed on the block:

1. Substitute each byte of the state with another value obtained from a table called S-box.
2. Rotate each row of the state to the left according to predefined values.
3. Perform matrix multiplication with each column of the state; this is called mix-column.
4. Finally, do XOR operation with the state and the round key.

AES is immune against differential and linear attacks. The reason for this is that the system design is based on Glorias Field (GF(2^8)) and uniform distribution of Sbox [10], [26], [30] and [32]. We strongly believe that in case of differential and linear attacks occur using AES, the long key size makes these attacks unfeasible. The same is true with DES22, where the longer key size makes differential, linear and Davies-Murphy attacks inapplicable to DES22.

The simple operations and product cipher used in DES and DES22 make them faster in execution than AES, since AES contains matrix multiplications.

8. CONCLUSION

What called us to go back to the past and search for what was used in the field of encryption, which was suitable for the speed of devices in that era, is the emergence of the so-called IoT. The world of the IoT may contain sensors with a simple processor that takes readings and sends them to the main processor, which in turn bases its decisions on these readings. It is essential to protect these readings as well as the speed of their transmission and processing. One of the factors that contribute to security and speed is the availability of an encryption algorithm that is fast and secure.

DES is faster than AES; however, it is not secure as AES. In this paper, DES is extended by increasing the key size and the permutation table is adapted to make the new product more secure and faster than both the original DES and AES. Extended DES is called DES22. The results of the experiment show that DES22 is faster and more secure than AES.

ACKNOWLEDGEMENTS

The authors would like to thank Yarmouk University for supporting them in conducting this research.

REFERENCES

[1] M. Barhoush, Persistent Protection in Multicast Content Delivery, PhD Thesis, Concordia University, Canada, 2011.

[2] M. Barhoush and J. W. Atwood, "Requirements for Enforcing Digital Rights Management in Multicast Content Distribution," Telecommunication Systems Journal, vol. 45, no. 1, pp. 3–20, DOI: 10.1007/s11235-009-9231-4, 2010.

[3] M. Michels, W. Fecke, J.-H. Feil, O. Musshoff, F. Lülfs-Baden and S. Krone, "Anytime, Anyplace, Anywhere'—A sample Selection Model of Mobile Internet Adoption in German Agriculture," Agribusiness, vol. 36, no. 2, pp. 192–207, 2020.

[4] M. E. Whitman and H. J. Mattord, Principles of Information Security, 6th Ed., Cengage Learning, 2017.

[5] A. Al Hayajneh, M. Z. A. Bhuiyan and I. McAndrew, "Improving Internet of Things (IoT) Security with Software-defined Networking (SDN)," Computers, vol. 9, no. 1, p. 8, DOI: 10.3390/computers9010008, 2020.

[6] R. Atkinson and S. Kent, "Security Architecture for the Internet Protocol," [Online], Available: https://datatracker.ietf.org/doc/html/rfc4301, 1995.

[7] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," IEEE Communications Surveys and Tutorials, vol. 19, no. 3, pp. 1628–1656, 2017.

[8] B. A. Forouzan, Cryptography and Network Security, Mc Graw Hill India, 3rd Edition, 2015.

[9] M. Agrawal and P. Mishra, "A Comparative Survey on Symmetric Key Encryption Techniques," International Journal of Computational Science and Engineering, vol. 4, no. 5, p. 877, 2012.

[10] W. Stallings, Cryptography and Network Security, 4th Edition, Pearson Education India, 2006.

[11] A. Pfitzmann et al., "More Efficient Software Implementations of (Generalized) DES," Computers & Security Journal, vol. 12, no. 5, pp. 477–500, 1993.

[12] E. Biham, "A Fast New DES Implementation in Software," Proc. of the International Workshop on Fast Software Encryption (FSE 1997), Part of the Lecture Notes in Computer Science Book Series, vol. 1267, pp. 260–272, 1997.

[13] R. Anderson, E. Biham and L. Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard," NIST AES Proposal, vol. 174, pp. 1–23, 1998.

[14] M. Naeemabadi, B. S. Ordoubadi, A. M. Dehnavi and K. Bahaadinbeigy, "Comparison of Serpent, Twofish and Rijndael Encryption Algorithms in Tele-ophthalmology System," Advances in Natural and Applied Sciences, vol. 9, no. 4, pp. 137–150, 2015.

[15] M. M. Alani, "DES96-improved DES Security," Proc. of the 7th IEEE International Multi-Conference on Systems, Signals and Devices, pp. 1–4, Amman, Jordan, 2010.

[16] M. Pranav and A. K. Rajan, "DES Security Enhancement with Dynamic Permutation," Proc. of the IEEE International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), pp. 6–11, Davangere, India, 2015.

[17] Y. Jun, L. Na and D. Jun, "A Design and Implementation of High-speed 3DES Algorithm System," Proc.

of the 2nd IEEE International Conference on Future Information Technology and Management Engineering, pp. 175–178, Sanya, China, 2009.

[18] D. Ma and Y. Shi, "A lightweight Encryption Algorithm for Edge Networks in Software-defined Industrial Internet of Things," Proc. of the 5th IEEE International Conference on Computer and Communications (ICCC), pp. 1489–1493, Chengdu, China, 2019.

[19] J. N. Mamvong, G. L. Goteng, B. Zhou and Y. Gao, "Efficient Security Algorithm for Power-constrained IoT Devices," IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5498–5509, 2020.

[20] M. M. Barhoush, N. A. Kofahi, K. M. O. Nahar, A. M. R. Alsobeh, A. Jaradat and B. Alomari, "Performance Enhancement of the Advanced Encryption Standard *via* Pipelined Implementation," Journal of Theoretical and Applied Information Technology, vol. 97, no. 15, pp. 4213–4226, 2019.

[21] S. Tillich and J. Großschädl, "Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors," Proc. of the 8th International Conference on Cryptographic Hardware and Embedded Systems (CHES'06), pp. 270–284, DOI: 10.1007/11894063_22, 2006.

[22] R. Sornalatha, N. Janakiraman, K. Balamurugan, A. K. Sivaraman, R. Vincent and A. Muralidhar, "FPGA Implementation of Protected Compact AES S--Box Using CQCG for Embedded Applications," Advances in Parallel Computing (Smart Intelligent Computing and Communication Technology), IOS Press, vol. 38, pp. 396-401, 2021.

[23] H. Zodpe and A. Sapkal, "An Efficient AES Implementation Using FPGA with Enhanced Security Features," Journal of King Saud University - Science, vol. 32, no. 2, pp. 115–122, 2020.

[24] C. Paar and J. Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Springer Science & Business Media, ISBN-13: 978-3642446498, 2009.

[25] U. S. N. B. of Standard, "Data Encryption Standard," Federal Information Processing Standards Publication, vol. 46, no. January 1977, pp. 1–18, 1977.

[26] B. A. Forouzan, Cryptography & Network Security, 1st Ed., McGraw-Hill, ISBN-13: 978-0073327532, 2007.

[27] C. E. Shannon, "Communication Theory of Secrecy Systems," Bell System Technical Journal, vol. 28, no. 4, pp. 656–715, 1949.

[28] D. Huertas, "DES Algorithm Implementation in C," DES/des.c, [Online], Available: https://github.com/dhuertas/DES, 2020.

[29] ProgrammerSought "AES Encryption Algorithm C++ Implementation," [Online], Available: https://www.programmersought.com/article/66314322796/.

[30] D. R. Stinson, Cryptography: Theory and Practice, 3rd Ed., Chapman and Hall/CRC, ISBN-13: 978-1584885085, 2005.

[31] S. Kunz-Jacques and F. Muller, "New Improvements of Davies-Murphy Cryptanalysis," Proc. of the 11th International Conference on Theory and Application of Cryptology and Information Security (ASIACRYPT'05), pp. 425–442, DOI: 10.1007/11593447_23, 2005.

[32] D. Ganguly, Cryptography and Network Security-Fundamentals and Practices, Mc Graw Hill Education (India), Private Limited New York, NY, ISBN 9781578087556, 2012.

ملخص البحث:

تقـدّم هـذه الورقـة إرشـاداتٍ كاملـة تتعلّـق بتكييـف نظـام (DES) وجعْلـه أكثـر أمانـاً، الـى جانـب تحسـين أدائـه مقارنـةً بـأداء خوارزميّـات التّشـفير الأخـرى مثـل (AES). إنّ الطّريقـة المقترحـة فـي هـذه الورقـة تزيـد مـن درجـة الأمـان فـي نظـام (DES) الأصـلي، عـن طريـق توسـيع حجـم المفتـاح دون التـأثير علـى تكلفـة النّظـام. الخوارزميـة الجديـدة تسـمّى (DES22)، وهـي ملائمـة للأجهـزة ذات القـدرة المنخفضـة علـى المعالجـة، مثـل المجسّـات اللاسـلكية. ولخوارزميـة (DES22) ثلاثـة خيـارات لحجـم المفتـاح: 128 و 256 و 512 بِـت (bits). كـذلك تقتـرح الورقـة تحسـيناً آخـر علـى نظـام (DES) مـن خـلال تبـديل الترتيـب علـى نحـوٍ عشـوائي وتوزيـع جـداول تبـديل الترتيـب الاسـتهلالية والنهائيـة بـين خوارزميـة التّشـفير وخوارزميـة إزالـة (فـكّ) التّشـفير. وتظهـر نتـائج التجربـة أنّ خوارزميـة (DES22) المقترحـة فـي هـذا البحـث هـي أكثـر أمانـاً وأسـرع من خوارزمية التّشفير المعتمدة على نظام (AES).