Jurnal Teknik Indonesia

Volume 5 Nomor 3 August 2026

https://jti.publicascientificsolution.com/index.php/rp

Overcoming DES Key-Length Limitations: A Novel Approach to Position Conversion and 128-Bit Key Encryption

Mochammad Imam Sulistyo Sarjowo[1*], Sugi Guritman[2], Eng Heru Sukoco[3], Irman Hermadi[4]

STT Wastukancana Purwakarta, Indonesia[1]

IPB University, Indonesia[2,3,4]

Email: imams@wastukancana.ac.id[1*], sugigu@apps.ipb.ac.id[2], hsrkom@apps.ipb.ac.id[3], irmanhermadi@apps.ipb.ac.id [4]

Keywords:

Information Security; DES; Cryptography

Abstract

The Data Encryption Standard (DES) is an algorithm designed to protect information by converting it from a readable form, known as plaintext, into an unreadable form, known as ciphertext. This process is called encryption, whereas the reverse process, which translates ciphertext back into plaintext, is referred to as decryption. The primary weakness of DES lies in its relatively short key length, which allows all possible keys to be exhaustively tested in a relatively short time. This research aims to address that weakness by designing a derivative of the DES algorithm with a key length exceeding 64 bits. The findings of this study indicate that the position conversion table within DES plays the most critical role in determining the key length that the algorithm can process. Consequently, this research focuses on developing a pattern for position conversion labeling, with the expectation that it can be utilized to construct a new position conversion table capable of handling keys longer than 64 bits. The final result of this study is the development of a new DES-derived algorithm with a 128-bit key, named DES-Aku. However, DES-Aku is not merely a derivative of DES; it also serves as a protocol for constructing position conversion labels within DES. As an algorithm, DES-Aku successfully performs character encryption using a 128-bit key, and as a protocol, DES-Aku successfully demonstrates a pattern for constructing position conversion tables in DES, which can further be utilized to develop new position conversion tables for keys exceeding 128 bits.

INTRODUCTION

According to the *Kamus Besar Bahasa Indonesia* (Great Dictionary of the Indonesian Language) (1991), security is defined as freedom from danger; thus, security is a condition that is very difficult to achieve, and the best that can be done is to minimize risk by reducing exposure to danger or threats. Cryptography, by definition according to the same dictionary (1991), is the study of secret codes. Thus, the science of cryptography is a technique for maintaining the confidentiality, or security, of information. The security concern of cryptography itself is typically directed toward data or services, rather than

the security of individuals or physical equipment (Gebremichael et al. 2020; Hedabou 2018; Martin 2020; Salman et al. 2018; Zou et al. 2016).

Messages in an intelligible form, or the original form of information, are referred to as plaintext, hereinafter written as *plaintext*; conversely, messages in an unintelligible form are referred to as ciphertext, hereinafter referred to as *ciphertext* (Guritman, 2003). The process of converting plaintext into ciphertext is called the encryption process, hereinafter referred to as *encryption*; conversely, the process of converting ciphertext back into plaintext is called the decryption process, hereinafter referred to as *decryption*. A variety of encryption algorithms have been developed, each with its own advantages and disadvantages. Some newer algorithms were created to address various shortcomings of their predecessors (Bakirci 2024; Hu et al. 2016; Swathi et al. 2024; Yuan et al. 2015).

DES (Data Encryption Standard) is an algorithm that was once highly prominent in cryptographic history, as it was declared secure by the U.S. government and adopted as an encryption standard by the NSA (National Security Agency) until restrictions were placed on disseminating its specifications abroad (Smid, 2021). However, the popularity of this algorithm declined sharply after it was successfully broken in 1998 (Ganesh et al., 2025).

The main drawback of the DES encryption algorithm is that its key length is too short, at only 56 bits. With this key length, brute-force attacks which attempt all possible key combinations become feasible within a short period of time (Alabdulrazzaq & Alenizi, 2022). This weakness was empirically demonstrated in 1998, when the entire DES keyspace was shown to be searchable in less than three days (Agate, 2023). Nevertheless, DES remains a compelling algorithm, born of sophisticated design principles that continue to invite admiration upon closer study (Wiemers, 2021). This is why many researchers have sought to remedy this flaw by increasing the length of the keys used (Alsuwaiedi & Rahma, 2023; Kareem & Rahma, 2020; Kareem & Rahma, 2022).

Several advanced algorithms and improvements to DES were subsequently developed. All of these derivative algorithms introduce new approaches to enable the use of keys longer than 56 bits. Among these DES-derived algorithms are 2DES, 3DES, NewDES, DES-X, and GDES. Of these, only 2DES and 3DES are generally considered to have successfully addressed DES's key-length weakness.

Although DES, 2DES, and 3DES all exhibit O(n) algorithmic complexity, test results indicate that 2DES requires twice the execution time of DES, while 3DES requires three times the execution time of DES.

With the exception of NewDES, all DES improvements leave the original structure of the DES algorithm unchanged. Modifying the DES algorithm by altering its underlying structure has only been attempted by NewDES; however, it has since been demonstrated that this approach yields a derivative algorithm no more secure than the original DES algorithm.

A 56- or 64-bit key length has been shown to be no longer viable for secure use; at the same time, the DES encryption algorithm is inherently unable to process keys of any

length other than 64 bits. While DES is known to lack sufficient key length to withstand brute-force attacks, existing DES-derived algorithms despite successfully resisting such attacks have introduced new problems of their own, as seen in 2DES and 3DES, which require two and three times the execution time of DES, respectively. Thus, it can be concluded that no existing derivative of the DES algorithm has fully resolved DES's weaknesses without introducing new limitations (Al-Hamadin 2021; Boateng 2022; Lin n.d.; Radhi 2023; Zheng et al. 2022).

This research aims to develop a method enabling DES to use keys longer than 64 bits specifically, 128 bits without incurring the significant performance degradation observed in 2DES or 3DES. The approach adopted in this study differs from previous DES improvement efforts by focusing on modifying the position conversion table, which is the primary determinant of the key length the algorithm can process. This research is also intended to serve as a guide for those seeking to implement the DES algorithm with a larger key length. Furthermore, this study aims to reveal the underlying formation pattern of the position conversion table in DES, thereby opening opportunities for the development of DES-derived algorithms with key lengths of n times 56 bits, without incurring an execution time n time longer than that of standard DES. Through this research, the formation pattern of the DES position conversion table will be identified, opening opportunities for the development of new DES-derived algorithms with key lengths n times longer than DES, without a corresponding n-fold increase in execution time. The novelty of this research lies in identifying the position conversion table as a key determinant of key length, as well as in developing a protocol for constructing new position conversion tables adaptable to various key lengths. This research is expected to contribute to the development of more secure and efficient cryptographic algorithms and to serve as a reference for future research in the fields of cryptography and information security.

METHOD

Frame of Mind

Various algorithms have been created to fix or mask the weaknesses of the DES algorithm, such as 2DES, 3DES, NewDES, DES-G and - DES-X, among others. Each algorithm has a different approach. Among all the algorithms above, only 3DES was considered the best until it briefly replaced DES's position as an encryption standard, until it was finally replaced by AES.

This research was made with the aim of improving, improving or covering the weaknesses of the DES encoding algorithm by using a method or approach that has never been done before. On the other hand, the final results of this study are expected to produce better algorithms than existing DES algorithm improvement algorithms, such as 3DES, 2DES or other derivative DES algorithms.

Behind its strong structure, it turns out that there are still loopholes that can be fixed in the DES algorithm, which have not been done by any repair algorithm. Some of these

gaps include increasing the size of the permutation innitial table, replacing the Sbox component, or changing the formula or equation used.

Materials and Tools

The equipment used in this study is two PCs (Personal Computers). One PC is used for program creation (coding), while the other is used for program testing. The computer used for the creation of the program has an Intel 2600 MHz processor, 512 Mb of RAM, and a Windows XP operating system.

The PC used for program testing has a 500 Mhz Celleron processor, 128 RAM, and a Windows Millennium Operating System. The application used in this study was created using the Microsoft Visual Basic 6 programming language.

Research Time and Place

This research was conducted at the IPB S2 Computer Laboratory. From March to May 2007.

Research Procedure

This research will begin by analyzing the DES encoding algorithm, in order to understand its strengths and weaknesses. In this early stage, any possible algorithm improvement opportunities will be noted to cover the main weaknesses of the DES encoding algorithm. The full research flow chart is outlined in the following Figure.

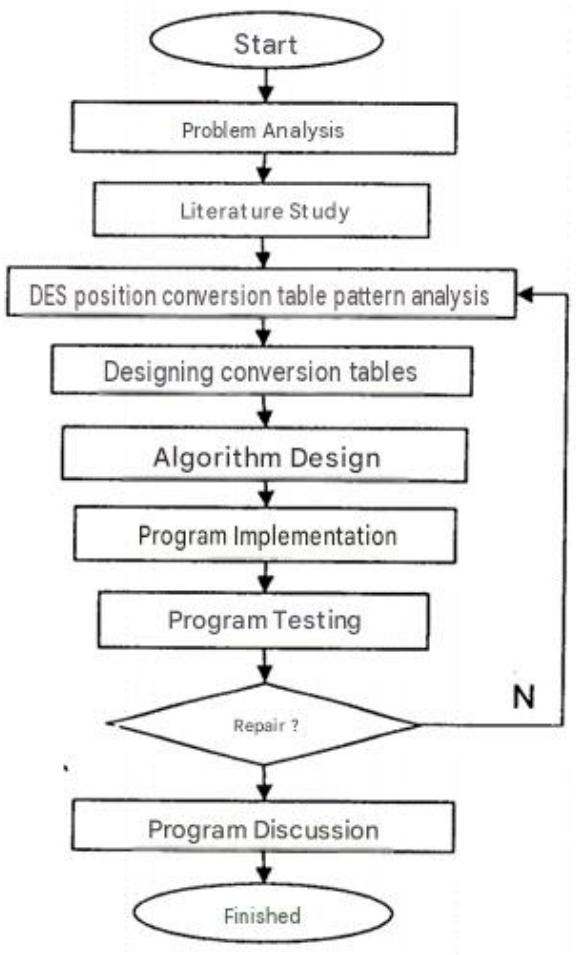

Figure 1. Research procedure flowchart

This research on improving DES performance is very closely related to literature studies, especially on the improvement of DES that has existed at this time. Various analyses must be carried out to find various weak points of the existing DES encoding improvements. A careful program design must also be carried out in order to produce an encoding algorithm that can improve the security of DES, and be able to provide better performance than existing DES repair algorithms at this time.

Measurement Parameters

Some of the parameters that will be used as a successful parameter for the design of this DES algorithm are the execution time, the complexity of the algorithm, and the opportunity to be penetrated by various attacks that are commonly carried out on the encryption algorithm.

RESULTS AND DISCUSSION

The DES encoding algorithm has been successfully implemented in the form of an expansion made with Microsoft Visual Basic 6.0. The purpose of compiling the code of this algorithm is to facilitate the next steps. As is known, the DES-Aku encoding algorithm will compare how it works with the 2DES encoding algorithm, which is basically a repeat of the DES algorithm twice.

The encoding application interfaces of both 2DES and DES-I have also been created, on the other hand the full line of code can be seen in the attachment. In order to work, the app requires a /ibrary COMDLG32.0CX file.

Implementation of the 2DES encoding algorithm.

The 2DES algorithm has been implemented in the form of an expansion, the goal is to compare its performance with the DES-Aku algorithm. The implementation of 2DES is a continuation or improvement of the program of the implementation of DES. 2DES is also implemented because it will be used as a performance comparison material against the DES-Aku algorithm made in this study.

Both 2DES and DES-I use 128-bit keys. For key purposes, this expansion has also been equipped with a password entry dialog box that requires users to enter a key of 16 characters.

Just like in DES implementations, the 2DES implementation also comes with a dialog box that can confirm the user if an encoding or translation process is successful, along with the time it takes to do so.

Implementation of the DES-Aku encoding algorithm.

1. Implementation of the DES-Aku Application

DES-Aku as an algorithm produced by this research has also been implemented in the form of expansion. The following image is an interface that is intentionally made to be the same as the 2DES interface.

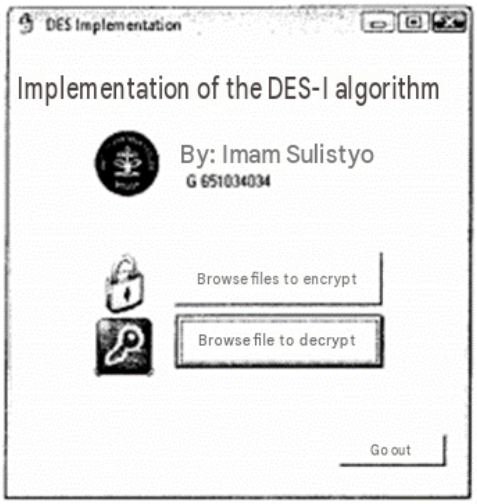

Figure 2. Display of the DES-Me interface

To carry out its function, this DES-Aku_ encoding application requires a 128-bit key and in practice this key is termed by the researcher as a 16-character password, which can be input by the user every time the user wants to encrypt. As described at the beginning, DES-I uses a 128-bit or 16-character key.

Here is a picture of the key input interface on the DES-Aku application.

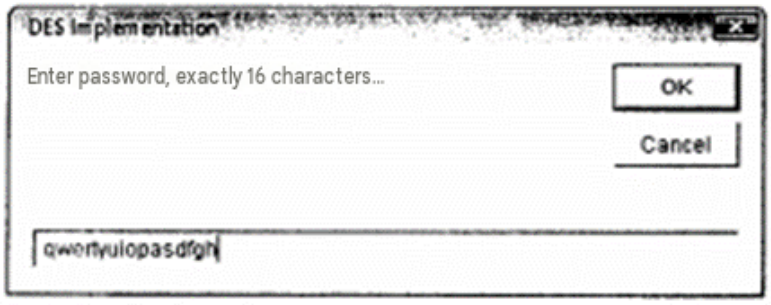

Figure 3. Enter the key

'inci Ae ie on Pat tsi PP SeBEENT A

sea Seo A TPL Seen em NSA AOS I BARE CMS A SEA SS RIE ORE EE et

The app has been tried on several file types and gives the best results when tried on text files. The following image is an example of a text file that has been tried in its original state, or before it was translated.

Figure 4. Example of plaintext

Once encoded, a new text file will be created, the content of which changes completely. The following image is what the message looks like after it has been encoded.

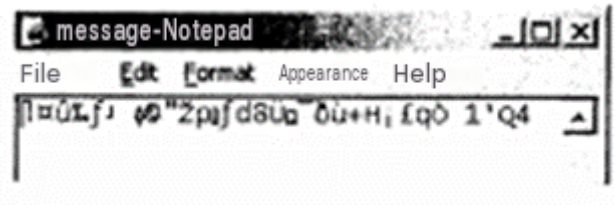

Figure 5. The resulting ciphertext

In order for communication between users and applications to run well, the researcher should not forget to create a dialog box that will explain to users if a message has been successfully encoded or translated, along with the time needed to do so. The following image shows what the app looks like when it notifies the user that the encoding or translation process has been successful, as well as the time it takes to encode or translate.

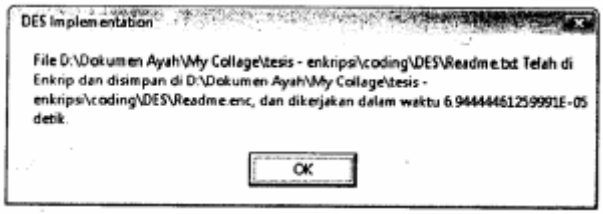

Figure 6. Encoding time measurement

2. Determination of the DES-Aku key

DES-I used a 128-bit key, twice as long as the original key used by the DES encoding algorithm. In DES-I there is no known bit parity, so the 128 bits of the key used are 128 bits of keys input by the user, although in practice these 128 bits of keys are entered by ~the user in the form of 16 characters.

DES-I'm using a 128-bit key length. This key length was chosen because it is enough with 128 key bits, it will result in 2'8 or 340,282,366,920,938,463,463,374,607,431,770,000,000 possibilities keys. Theoretically, the probability of this key combination is already considerably greater than the probability of a DES key combination that has only 56 key bits, which yields only 2°° or 72,057,594,037,927,936 possible keys.

The length of the DES-I key is the same length as the 2DES key which also yields 2' or 340,282,366,920,938,463,463,374,607,431,770,000,000 possible keys. In terms of security, the DES-Aku key is as strong as the existing DES repair algorithm, namely 2DES.

The purpose of making a 128-bit key is for the convenience of the user of the DES-Aku software. By using a 128-bit key, the user only needs to memorize 16 characters, on the other hand, if using a key that is 192 bits or more, the user must memorize the key at least 24 characters. Memorizing 16 characters will of course be much easier than memorizing 24 characters, because there is a human error factor that can occur when the user memorizes or uses the encoding key

If needed in the development of the DES-I expansion, it can have keys longer than 128 bits, such as 256, 512 or even 1024 bits. The use of longer keys is still possible

because the content of this thesis in addition to creating an encryption algorithm with a key length of 128 bits, this thesis also reveals the pattern of forming position conversion tables in DES, so that position conversion tables can be easily recreated in the future with even larger key lengths.

3. Discussion of Functions

The f function used in DES-I is the same as the f function used in the DES algorithm. The function that is executed in each round in this Feistel permutation has the following mathematical equation:

$$L_i = f(R_{i-1}, Ki) = P[S\{E(R_{i-1}) \oplus K_i\}]$$

It can be seen that this function aims to fill the left segment of a Feisel permutation round, which is the result of a function that involves the right segment of the previous round and the specific subkey on that rode. Some of the operations involved in this function are permutation, substitution, expansion and XOR operations. Expansion is an operation inside the function f that is performed at the earliest. This operation aims to prepare the data block to be encoded, because the data block is only 64 bits in size, on the other hand the sub key used is 96 bits. This operation will increase the size of the data block to be encoded from 64 bits to 96 bits, thus equal to the number of bits of the sub key used.

The next operation in the DES structure is the XOR operation, which will combine the expanded data blocks contained in a specific subkey in each round. The XOR operation works by "justifying" two different data, and "blaming" the same two data. XOR operations are logical operators that have reversible properties; thus, they are often used in various encoding and translation algorithms.

A XOR B → C,
A XOR C → B,
B XOR C → A.

The next operation in the f function is substitution. This operation serves to compress the 96-bit XOR data block back to its original size of 64 bits. The substitution operation works by dividing a 96-bit block of data into 16 subblocks that are each 6 bits in size. Each subblock is then substituted against the Sbox Table until it becomes a new 4-bit data subblock. The discussion of XOR is very closely related to the discussion of the Sbox Table as listed in the next few sub-discussions.

The last operation in the f function is the Feisel permutation operation. The last step of the int is really just a step to change the order of the table (location transposition). with the aim of increasing the complexity of the encoding algorithm and further randomizing the data. There is no change in the amount of data in the permutations of this table.

4. IP and IP Table Discussion"!

The IP table and its inverse (IP"') is a position conversion table used by the DES algorithm. The arrangement of numbers in the IP Table can actually be freely composed by anyone and does not have to follow the one provided by the DES, it just has to follow the same dimensions, and the IP must be obtained from the corresponding IP Table.

On the DES algorithm, an IP Table is visible! This can be seen from the pattern that is more noticeable in the IP position conversion table, rather than the pattern shown in the IP table, as shown in Figure 17.

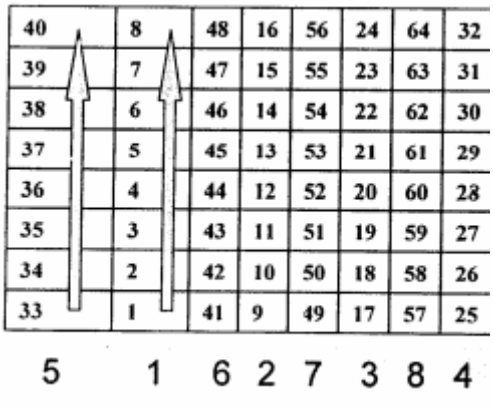

Figure 7. IP Table Patterns on DES

It can be seen that the IP Table which is composed of 8 columns begins to be filled in the first even column, which is the second column, from the last row to the first row. The filling in of numbers then continues in the next even column, in a fixed order from bottom to top. After all even columns are filled, only the odd column is filled. 4

The IP table on the DES is then obtained as a result of the position conversion performed by the IP Table'. There is no numerical pattern in this table, thus the final shape of the IP Table for the DES algorithm is as shown in the image above.

The pattern seen on the 'IP Table', can be easily followed by the DES-Me algorithm. thus, the process of creating IP Tables"! which is followed by the IP table on DES-I can be easily compiled.

The [P' and IP Table components in DES-I have the same pattern as the "IP" and IP Tables in the style of DES, only the components used here are larger. The DES algorithm uses a table with 64 components; thus, the DES-I algorithm uses a table with 128 components. IP Table"! The DES-Aku algorithm used is as follows:

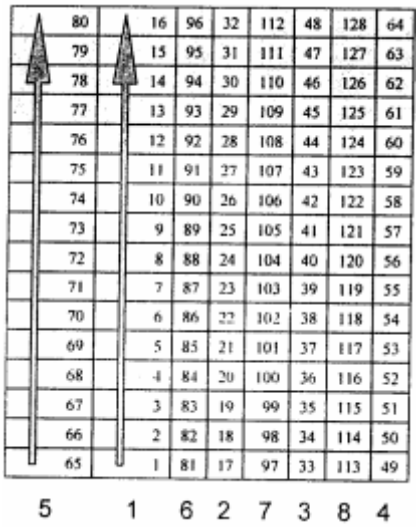

Figure 8. IP Table Patterns on DES-Me

On the other hand, the IP Table of DES-*Aku* itself is only the result of a positional conversion of the original DES 'IP Table,' without any definite underlying pattern. The final form of the IP Table for the DES-*Aku* algorithm is presented in Table 10.

5. PC1 Table Discussion

PC1 is a table used to compress keys. In the DES algorithm, this table functions to convert a 64-bit key back to a 56-bit key; in contrast, in DES-*Aku*, this table functions to convert a 128-bit key to 112 bits. The PC1 variables in DES-*Aku* were also created with attention to the principles and patterns used to construct the PC1 Table in the DES algorithm.

The working principle of this table-compression process is to remove certain bits by repositioning the table components. The end result is a new table with a new arrangement that has lost some of its original components, giving the final table smaller dimensions.

In fact, there is nothing particularly special about this table, and thus, in the design of DES-*Aku*, such tables can be easily replicated. The PC table is only one of two tables that play a role in the key derivation process and does not directly determine the form of the ciphertext generated from the plaintext.

The PC1 table of the DES-*Aku* algorithm is created by omitting 16 numbers from a sequence of 128 numbers. These sixteen numbers can be chosen at random. The remaining 112 numbers are then arranged randomly, thus forming a new PC1 Table for the DES-*Aku* algorithm. The final form of the PC1 Table for the DES-*Aku* algorithm is shown in Table 12.

6. PC2 Table Discussion

The PC2 table is one of the two tables that play a role in the key derivation process. Like the PC1 Table, the PC2 Table also does not directly determine the form of the generated ciphertext, and so it can likewise be easily replicated. By eliminating 8 random numbers from the 112 numbers *[note: one line of the original source is corrupted/illegible and could not be reliably reconstructed]* and then randomizing the remainder freely, a

PC2 Table for the DES-*Aku* algorithm is obtained. The final form of the PC2 Table for the DES-*Aku* algorithm is shown in Table 13.

7. Expansion Table Discussion

The expansion table is one of the tables involved in the Feistel function and plays a very important role in determining the form of the resulting ciphertext. In the DES algorithm, this table functions to expand a 32-bit data subblock to 48 bits. This 48-bit size corresponds to the size of the subkey used in each iteration of the Feistel operation; having matching sizes enables the XOR operation between data subblocks and specific subkeys at each iteration. In the DES-*Aku* algorithm, the expansion table serves the same function, except that the size is changed from 64 bits to 96 bits.

Expansion tables have a pattern that is very easy to identify (Table 5). This highly visible pattern makes the DES expansion table easy to replicate and extend (Table 14). The underlying principle of the Expansion Table is to represent the table in the form 4 × (key length / 8). New columns are then added to the left and right of each table, with component values continued sequentially from one side to the other.

8. Discussion of the S-box Table

The S-box table is the primary security mechanism in the DES algorithm. This table functions as a one-way operation, meaning that an output value cannot be reverted to its original constituent input values. The S-box table in DES-*Aku* consists of 16 columns and 64 rows, with each row containing values arranged randomly from 0 to 15; no two rows share the same numerical sequence. The S-box table used by the DES-*Aku* encryption algorithm is presented in Table 16.

The function of the S-box Table is the inverse of the Expansion Table: this table compresses the size of the XOR-processed data subblocks (combined with specific subkeys) back to their original size. In DES, this table reduces data size from 48 bits to 32 bits; in DES-*Aku*, this table reduces data size from 96 bits to 32 bits.

S-box tables function by substituting 6-bit data segments with 4-bit data segments. To accomplish this, the data to be compressed must first be broken down into subblocks of 6 bits each. The DES algorithm divides the data into 8 subblocks, whereas the DES-*Aku* algorithm divides the data into 16 subblocks.

The S-box itself is a two-dimensional table consisting of rows and columns. The original 6-bit data segment is divided into two parts: the 1st and 6th bits are combined and converted to decimal to serve as the row reference, while the 2nd through 5th bits are combined and used as the column reference. The value at the intersection of the corresponding row and column becomes the new data, which, when converted to binary, forms a 4-bit output. The pattern of the S-box Table follows a random arrangement of 16 numbers per row, with no two rows sharing the same sequence.

9. Discussion of the Feistel Permutation Table

The permutation table has no effect on data size; its sole function is to randomize the arrangement of the data array. This table represents the final step of the Feistel function. In DES, this table is 32 bits in size; in DES-*Aku*, it is 64 bits. The Feistel permutation table used by the DES-*Aku* encryption algorithm is presented in Table 15.

Nonetheless, the Feistel permutation table is the most difficult table for which to discern a pattern, owing to its randomized numerical arrangement. After exploring various possibilities, the researchers' solution was to construct a new table, 32 bits in size, whose numerical arrangement matches the original DES permutation table exactly, with 32 added to each component value. The original DES table was then appended to this new table, expanding its total size to 64 bits. This resulting table is used by DES-*Aku* as the Feistel permutation table incorporated into the Feistel function.

Performance Test Results Between DES-Aku and 2DES

Different algorithms naturally yield different levels of performance; a superior algorithm will demonstrate superior performance. The performance parameters measured in this study include algorithm complexity, execution time, and security testing.

1. Algorithm Time Complexity Analysis

The algorithm was implemented using Microsoft Visual Basic 6.0, a widely used programming language. Visual Basic is a high-level programming language equipped with a variety of ready-to-use objects designed to facilitate ease of use for developers.

These objects include, among others, command buttons and list boxes. However, because Microsoft Visual Basic 6.0 operates as a "black box," users are unable to view the underlying script execution. For this reason, the analysis of the algorithm's time complexity could not be conducted based on the script code itself, but only at the algorithmic level.

From beginning to end, the DES-*Aku* algorithm consistently involves a looping process, beginning with the segmentation of the message to be processed into 128-bit blocks. The number of repetitions depends on the length of the message; the longer the message, the more repetition processes occur. The complete flowchart for the DES-*Aku* process is provided on the CD accompanying this thesis.

From the flowchart, it can be observed that from the 5th to the 11th row, there are 16 looping processes, with 5 sub-processes occurring within each loop. The resulting complexity of this loop is $O(16 \times 5)$.

The second source of complexity is observed from the 14th to the 24th row, where the length of repetition depends on the length of the message, denoted N. Each loop contains 7 processing steps, and rows 17 through 19 additionally contain a sub-loop that repeats 16 times, each containing 1 processing step. Thus, the complexity becomes:

The final complexity result is $O(23N + 80)$. For simplification, this equation can be reduced to $O(N)$. This complexity equation applies equally to both DES and DES-*Aku*. In 3DES, steps 14 through 25 are performed twice, so the complexity becomes $O(2 \times 23N + 80)$, which can likewise be simplified to $O(N)$.

These results demonstrate that DES, DES-*Aku*, and 2DES all share the same complexity order, $O(N)$, with message length (N) being the only factor influencing the relative encryption speed of DES, DES-*Aku*, or 2DES. In practice, however, the execution speeds of the three algorithms are not identical; a measurable difference exists even at the

scale of 1/100,000 of a second. This difference in execution time is discussed further in the following subsection.

2. Algorithm Complexity Test

An algorithm is a set of instructions for solving a problem. Algorithm complexity is a measure of the computational resources an algorithm requires to solve a given problem. The results of the complexity testing for the DES-*Aku* and 2DES algorithms are presented in the following figure.

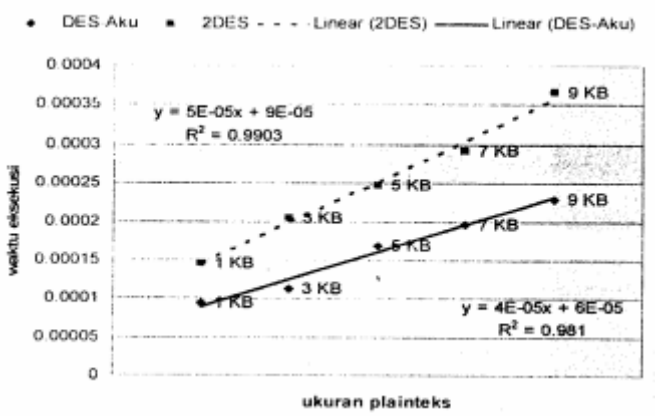

Figure 9. Comparison of the DES-Aku performance test against 2DES

From Figure 19 it can be seen that DES-Aku has inherited one of the advantages of DES, which is that behind its complex structure, DES-Aku has a low complexity, so that the time of execution depends only on the size of the plaintext, with a direct relation to it. The results of the above measurement have been strengthened by the high level of confidence of the trend line above.

The results of the algorithm complexity test on DES-I show the result of O(n). These results have been tested on several plaintexts of different sizes, but performed on the same machine. For each measure, several repetitions were performed, while the final result was the average of each different plaintext measure.

Figure 19 has also shown that both 2DES and DES-Aku have the complexity of the O(n) algorithm, but DES-Aku is still superior because it has a rendah_ gradient so that when implemented with the same data, DES-Aku will definitely be able to run faster.

1. Run Time Test

Execution time is the time it takes to run a program. The execution time of a program is closely related to the specifications of the computer used, especially speed and memory issues. Another factor that is also involved in execution time is the state of the computer. If at the same time the computer used to perform the test turns out to be performing other operations in parallel, the measurement maka_hasil may be different.

The execution time test was carried out as proof of the results of the algorithm complexity test. The computer used to test the execution time of the three programs involved in the int thesis had a speed of 500 Mhz and 128 MB of RAM. Time sampling was carried out as many as 10 repetitions. In each measurement, different results were

obtained, until in the end an average was carried out. The results of the execution time measurement of the 3 algorithms involved are presented in the table below:

Table 1. DES-Aku Execution Time Test against 2DES

Nama algoritma	Waktu eksekusi
2DES	0.0001123 detik
DES-Aku	0.0000787 detik

The table above has proven that DES-Aku is indeed faster than 2DES. The DES-I speed difference is 1.4 times faster than the 2DES execution time. Although the time difference between the two is very thin, this time difference will be sharper and more significant with the length of the encoded plaintext, so that if the encoded plaintext is very large, then the difference in execution time between the two will be more visible.

2. Safety Test

In accordance with the original research objective, the ciphertext generated by the DES-Aku encoding process must be different from the plaintext. Figure 14 is an example of the original plaintext, and Figure 15 is an example of the ciphertext of the encryption.

Based on the two images, it can be seen that the images resulting from the encoding process are very random and very different from the original images. With this difference, it is expected that plain text cannot be guessed if only by looking at the sifertext.

If a cryptographer tries to carry out an attack by relying only on knowledge of the frequency of occurrence of a character (Table 8), such as the very frequent occurrence of the letter 'a' in Indonesian, then this would be impossible, because the DES-Aku encoding process will not display the same ciphertext character for the occurrence of the same plaintext character multiple times. The resulting ciphertext character does not depend solely on the plaintext character, but also depends on the entire DES-I component, such as the position conversion table used, or the specific subkeys involved in each round.

If the cryptographer has the corresponding plaintext and sifertext at the same time. Then he uses the two images to carry out a known-plaintext attack to find out the subkey used, so he will encounter many obstacles. Because plaintext is not just carried out XOR operations with a row of key flows. However, each bit in the plaintext must first be shuffled in position using the IP Table, followed by the process of expanding the data block, then performing an XOR operation with a row of key flows.

The result of this XOR is then substituted again with the Sbox Table, on the other hand the Sbox Table is a one-way function replacement, which makes the process of reversing the ciphertext algorithm into the original plaintext very difficult to do, because for each ciphertext character to be searched, the cryptographer will face 16 possible plaintexts. Moreover, the result of this Sbox substitution in the end had to be re-arranged with a Feistel permutation. The end result of this series of safeguards is what makes the process of algoritnia reversal impossible to do

The last possible attack that a cryptographer can perform is to try all the possible brute force attacks. Although this is very possible, the key space that must be solved is

very large. This can be seen from the length of the DES-I key which is 128 bits, which results in 2128 possible keys. Assuming that the DES algorithm takes 3 days to trace all the keys. With a key length of 56 bits, the DES algorithm has 256 key spaces. From the data, it can be seen that DES-I takes $\frac{2^{128}}{2^{56}}$ x3 days or 1.4 x 10 days or 2.8 x 1019 years.

Although computing capabilities are now 30 times faster than when the DES algorithm was successfully solved, it will take at least 1.2 x 108 years for DES-Aku to be able to test all its key opportunities.

The overall power of the DES-Aku outlined above is also possessed by the 2DES encoding algorithm. However, the 2DES encoding algorithm has serious problems. i.e. very weak to the MITM (Meet in The Middle) type of attack. This weakness is due to the repetition of the DES algorithm evenly, thus it is enough to have a known plain text, and the attack attempts all possible keys are carried out in terms of both ciphertext and plaintext, then if the result between the two encoding executed in series is the same, then that is the key used. This weakness of 2DES is not possessed by DES-Aku, and this factor makes DES-Aku superior to 2DES.

If. Cryptologists try to attack with differential cryptologists serangan_, simply by being guided by the strength of the DES in dealing with this attack, the DES-Aku will of course inherit the same power. Differential cryptographic attacks have been around since 1974, long before DES was created, so DES designers have anticipated these attacks using the Sbox Table.

After all, differential cryptanism requires 2" pairs of known sifertext plaintexts. In fact, to find 2 pairs of sifertcks plaintext for the same key is very difficult, to the point that this weakness would only exist theoretically. A summary of the performance test between DES-Aku and 2DES is presented below.

Table 2. Summary of the results of the DES-Aku test against 2DES

Kriteria	2DES	DES-Aku	Seimbang
Algorithm complexity		✓	
Execution time		✓	
Bruteforce attack	✓		
Letter frequency attack			✓
Differential analysis attack			✓
Linear analysis attack			✓
MITM attack		✓	

CONCLUSION

This study concludes that the DES-Aku algorithm is a development of the DES algorithm that is able to overcome the limitation of key length by modifying the position conversion table so as to support the use of 128-bit keys. This development provides a better level of security than 2DES, as it has no weaknesses against Man-in-the-Middle attacks and has faster execution times although the duration of the process increases as

the length of the encrypted message increases. In addition, the DES-I concept opens up opportunities for the development of algorithms with larger key lengths, such as 512-bit or 1024-bit. Therefore, further research is recommended to examine the relationship between the increase in the length of the key space and the execution time in order to determine the most optimal configuration based on security aspects, computing efficiency, and ease of use.

REFERENCES

Al-Hamadin, Rashed J. 2021. "A New Approach for Data Symmetric Key Cryptography Using Fast Neural Networks with Single Step of Backpropagation and Finite Fields."

Agate, V. (2023). Bayesian modeling for differential cryptanalysis of block ciphers: A DES instance. *IEEE Access, 11*, 4809–4820. https://doi.org/10.1109/ACCESS.2023.3236424

Alabdulrazzaq, H., & Alenizi, M. N. (2022). Performance evaluation of cryptographic algorithms: DES, 3DES, Blowfish, Twofish, and Threefish. *International Journal of Communication Networks and Information Security (IJCNIS), 14*(1). https://doi.org/10.17762/ijcnis.v14i1.5262

Al-Hamadin, R. J. (2021). *A new approach for data symmetric key cryptography using fast neural networks with single step of backpropagation and finite fields*.

Alsuwaiedi, H. K. A., & Rahma, A. M. S. (2023). A new modified DES algorithm based on the development of binary encryption functions. *Journal of King Saud University: Computer and Information Sciences, 35*(8), Article 101716. https://doi.org/10.1016/j.jksuci.2023.101716

Bakirci, M. (2024). Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems (ITS) applications. *Digital Signal Processing, 152*, 104594.

Boateng, I. D. (2022). A critical review of emerging hydrophobic deep eutectic solvents' applications in food chemistry: Trends and opportunities. *Journal of Agricultural and Food Chemistry, 70*(38), 11860–11879.

Ganesh, R., Khan, B. U. I., Khan, A. R., & Kamsin, A. B. (2025). A panoramic survey of the advanced encryption standard: From architecture to security analysis, key management, real-world applications, and post-quantum challenges. *International Journal of Information Security*. https://doi.org/10.1007/s10207-025-01116-x

Gebremichael, T., et al. (2020). Security and privacy in the industrial Internet of Things: Current standards and future challenges. *IEEE Access, 8*, 152351–152366.

Guritman, S. (2003). *Diktat pengantar kriptografi*. Fakultas Matematika dan IPA, Institut Pertanian Bogor.

Hedabou, M. (2018). Cryptography for addressing cloud computing security, privacy, and trust issues. In *Computer and cyber security* (pp. 281–304).

Hu, A., Noble, W. S., & Wolf-Yadlin, A. (2016). Technical advances in proteomics: New developments in data-independent acquisition. *F1000Research, 5*, F1000 Faculty.

Kareem, S. M., & Rahma, A. M. S. (2020). New modification on Feistel DES algorithm based on multi-level keys. *International Journal of Electrical and Computer Engineering (IJECE), 10*(3), 3125–3135. https://doi.org/10.11591/ijece.v10i3.pp3125-3135

Kareem, S. M., & Rahma, A. M. S. (2022). Development of Data Encryption Standard

algorithm based on magic square. *Indonesian Journal of Electrical Engineering and Computer Science, 28*(1), 297–305. https://doi.org/10.11591/ijeecs.v28.i1.pp297-305

Lin, F. (n.d.). *Transforming district energy systems for sustainable heating and cooling through renewable integration*. SSRN. https://ssrn.com/abstract=6016794

Martin, K. (2020). *Cryptography: The key to digital security, how it works, and why it matters*. W. W. Norton & Company.

Radhi, S. M. (2023). In-depth assessment of cryptographic algorithms namely DES, 3DES, AES, RSA, and Blowfish. *Iraqi Journal of Computers, Communications, Control and Systems Engineering, 23*(3), 11.

Salman, T., et al. (2018). Security services using blockchains: A state of the art survey. *IEEE Communications Surveys & Tutorials, 21*(1), 858–880.

Smid, M. E. (2021). Development of the Advanced Encryption Standard. *Journal of Research of the National Institute of Standards and Technology, 126*, Article 126024. https://doi.org/10.6028/jres.126.024

Swathi, Y., & Challa, M. (2024). YOLOv8: Advancements and innovations in object detection. In *International Conference on Smart Computing and Communication* (pp. 1–13). Springer.

Wiemers, A. (2021). Improving recent side-channel attacks against the DES key schedule. *Journal of Cryptographic Engineering*. https://doi.org/10.1007/s13389-021-00279-2

Yuan, Y., et al. (2015). Balancing convergence and diversity in decomposition-based many-objective optimizers. *IEEE Transactions on Evolutionary Computation, 20*(2), 180–198.

Zheng, L., Wang, Z., & Tian, S. (2022). Comparative study on electrocardiogram encryption using elliptic curves cryptography and Data Encryption Standard for applications in Internet of Medical Things. *Concurrency and Computation: Practice and Experience, 34*(9), e5776.

Zou, Y., Zhu, J., Wang, X., & Hanzo, L. (2016). A survey on wireless security: Technical challenges, recent advances, and future trends. *Proceedings of the IEEE, 104*(9), 1727–1765.