DES
Implementation and Demonstration

Nduvho Ramashia
1490804

April 2022

1 Introduction

This document details the implementation and demonstration of the Data Encryption Standard (DES) cipher, which is based on the Data Encryption Algorithm (DEA). The DES is a symmetric 56 bits key and 64 bits massage block cipher.[Cheng] It was first standardized in 1975 [1]. It has since become less used due to its relatively short 56 bits key length. [Chris] Even though it is not used as much for high security data, it is however still in use in some areas. Besides that, its version, the triple DES, is still considered considerably safe to this day due to its $2^(3 * 56)$ key-space.[2]

It is also known that for a single DES, as opposed to a triple DES (i.e. 3DES), there are a some keys that are considered weak, some semi-weak, and some possibly weak. The total number of these keys however is significantly lower than the overall possible keys, which is why the fact of their existence is usually not seen as major flow of the algorithm. [2]

Overall, from afar, the DES algorithm consist of: – The initial permutation – The 16 rounds that involves the fiestal function – The inverse of the initial permutation

Section 2 details the implementation of the key scheduling function, which is needed by the fiestel function during the 16 rounds. After that, in section 3, it is then demostrated using the predefined keys. Using the function, the encryption key is used to produce needed subkeys. The subkeys are evaluated to help with the key classification. That meaning that if all the subkeys are idential, then the encryption key falls under the class of weak keys. Section 4 details the implementation of the function that performs the 16 rounds of the DES. Later in section 5 is where everything is brought together to join the pieces that make up the fully functioning DES algorithm. Section 6 details the

Key	Class	Motivation/Reason
1F1F1F1F0E0E0E0E	Weak	Only 1 unique subkey
1FFE1FFE0EFE0EFE	Not weak	12 unique subkeys

Table 1: Key Classes

results and analysis of the implemented DES algorithm. Section 7 point out the design challenges faced as well as future recommendation and improvements on the implementation.

2 Algorithm Implementation

2.1 The Key Scheduling function

The algorithm takes the 56 bits key and use it to create sixteen 48 bits keys, commonly refered to as subkeys. Each of the 16 roubds of the algorithm make use of one of these keys.

A function *subkeys_gen* was used to generate these. It was created to take as input a 64 bits key, which it then accordingly process to get the 56 bits key and then compute all the subkeys for the algorithm and the given key.

To create the keys for each round, up to the total of 16 rounds, it uses another previously created function *subkey_gen*. This function takes an its input a 56 bits key and then return a sukey.

The *subkey_gen* function was also used to detemine the classify the 64 bits keys:

- 1F1F1F1F0E0E0E0E
- 1FFE1FFE0EFE0EFE
- 1FFEFE1F0EFEFE0E

since it was defined to also compute the key proparties that make it possible to figure the class of a key.

The results of the above keys classification can be found in table tab:keyclasses

The keys were provided in the little endian format, nibble-wise. They were then reassembled to have the most significant bit (MSB) on the far left and the least significant bit (LBS) on the far right.

2.2 The DES Round function

After implementing the sub-keys generating function, the DES round performing function was written. The function takes as its input a 64 bits massage block, a round number and sub-key for the round. It then after doing all the necessary permutation and additions, returns two 32 bits long blocks.

2.3 The overall DES algorithm.

After writing functions for all the necessary middle steps, they were all brought together into one function *DES_1490804*. This function combined all the parts to form a fully functioning DES algorithm. It was written to take a two 16 characters, representing 16 hexadecimal numbers, which are the equivalent to two 64 bits binary numbers. It was written to take them that way to make it more convenient for users to or text their massages and keys manually, using a relatively more readable format massage and keys, expressed in hexadecimal as compared to a 64 bits long number.

The function was written to then pre-process its input accordingly to make them ready for the functions it makes use of. Figure fig:DES shows the patterns that the DES functions uses to encrypt the 64 bits massages using the also given 64 bits keys. For organisation purposes, the each function was put on its only function file.

3 Results and Testing.

To test the overall algorithm implementation, a number of online DES encrypting sources were used, to validate its encryption. Those include: — — — —

(time test) (multiple inputs speed tests)

4 Challenges faced and future recommendation or possible improvements

The functions implementation can use predefined and optimised functions to improve the functions' overall speed. That is provided the overall code readability is not severely affected Challenges faced include data compatibility between functions. Meaning some functions had to be readjusted to better fit the data type that the other functions that depend on them or on which they depend. This was due to the fact that some data types could not be easily manipulated by the dependent functions, which prompted the data type on the functions proving the data. First planning the

overall algorithm to make sure the data types return and provided by each function is compatible is recommended. That meaning taking a step back and took at the algorithm as a whole.

5 Conclusion

The DES algorithm was implemented successfully. The overall implementation was done using a total of four functions, namely 'subkey_gen', 'subkeys_gen', 'des_round' and the main function 'DES_1490804', which were all on their only separate function files, for organisation purposes.

References

[1] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners.* Springer Science & Business Media, 2009.

[2] L. Cheng, "Modern symmetric ciphers," March 2022.