Authors: Jayson Rook and Zachary Zampa
Date: 12/12/2018
Class: ECE 287
Topic: Triple DES Encryption

Description of Topic:

The purpose the project is to utilize the FPGA board to conduct Triple DES Encryption. The user can enter hexadecimal values through a keyboard or switches on the board. Once 64bits have been entered, the board will run the encryption algorithm and display the encrypted value on the second line.

Background Information:

The DES algorithm, as explained here[1], takes a 64 bit message and converts it into a string of 64 encrypted bits, using a 64-bit key (of which only 56 are actually used). In order for decryption to be possible, all the operations must not lose information - so the algorithm uses only permutations (swapping the locations of bits) and XOR operations, rather than gates like AND or OR which do not allow an input to be deduced given the output and the other input. Triple DES, which we have implemented, actually performs 3 steps:

1. A DES encryption with a key A
2. A DES decryption of the output of step 1, using a different key B
3. A DES encryption of the output of step 2, again using key A

While DES is now no longer considered secure (it is too easy to crack), triple DES is much more secure, and it is unlikely to be cracked in the near future.

The difference between encryption and decryption in DES is very slight, in terms of programming. The algorithm transforms the original 64-bit key into 16, 48-bit “subkeys” for use in the 16 main iterations - decrypting a message is achieved simply by reversing the order these subkeys are used.

Layman's Description of Design:

The design works through a user entering values on a keyboard. The programmed board takes these inputs and adjusts where it is inputed accordingly downward. Once the entire input is entered, the board runs through the DES Encryption Process and outputs the result. The encryption process has multiple steps. It begins with swapping the initially inputted bits, based on a set algorithm. Afterwards, the input goes through a series of 16 “rounds”. In any specified round the key gets shifted one bit down slightly. Then, the right half of the message gets expanded from 32 bits to 48 bits by repeating the first and fourth bit of a block on each side. Next, the expanded input is substituted with set substitution values and reduced back to 32 bits. The right half of the message then gets permuted again and swapped with the left half of the message through a process called XOR (which is where the left message half and right half get combined where if two 1s combine a 0 forms, if two 0s combine a 0 forms, and if a 1 and a 0 combine, a 1 forms). This “round” process repeats a total of 15 more times after the initial. At the end of the rounds, a final permutation occurs and the right and left halves are combined into one full block again. This combined output is the encrypted block. In Triple DES Encryption, this process is repeated two more times. In this, the first stage is done through the first specified key. In the second stage, the encrypted message is decrypted with a second key. This second key is different and will not reproduce the same result as the initial key decrypting. In the third stage, the “decrypted” message is encrypted with the first key again. To decrypt, the entire process is the same, except for the substitution stage, which is done in reverse.

Modular Design

The project is implemented in 10 Verilog (.v) files:

- ECE287_Project.v - the top-level entity, which instantiates modules for getting input (Bit_Input) and displaying to screen (LCD_Display)

- LCD_Input.v - contains the LCD_Display module, which takes a clock, a screenRST (which tells the screen when to refresh), and two buses (inHexChars and outHexChars) which contain the character codes for everything the board should display

- Bit_Input.v - invokes the keyboard module. This module also allows a user to input the values either through FPGA switches or a PS2 type keyboard.

- Bit_Converter.v - converts a group of 4 bits into the character code for writing the hexadecimal character on the screen

- keyboard.v - takes pressed buttons on the PS2 type keyboard and converts them to language the FPGA board can read.

- PS2_Controller.v - Facilitates communication between a PS2 type keyboard and the FPGA board.

- Triple_DES.v - instantiates the DES encryption modules and ensures the correct key and message values gets transmitted to the corresponding DES module

- DES_Encrypter.v - the actual DES algorithm, taking the input and key and generating an output. This module is implemented combinationally, but its output is only displayed on the screen once the user has finished inputting all 16 hexadecimal characters

- Subkey_Generator.v - takes the original 64-bit key, and generates the 16, 48-bit subkeys needed for the main DES algorithm

- fFunction.v - implements the Feistel function, an intermediate step used in each round of the DES encryption

Conclusion (Summary of Project and TLDR of Report):

In short, this project allows for a user to encrypt 64 bits of data with Triple DES levels of encryption. The user can also decrypt an encrypted value on the board as well. Part of the security of the board is the key being hidden in the board. This means that one must use a similarly programmed board to decrypt any message received.

How to use

A video of the system in use can be found here[2]. The important steps are:

1. Turn reset switch on (SW17)
2. Specify encryption or decryption (SW15)
    a. High is decryption, low is encryption
3. Input message: 2 options
    a. Keyboard
        i. Type hexadecimal digits to input (0-9, a-f)
        ii. Backspace removes the last entered value
        iii. Delete clears all input
    b. DE2 Switches
        i. Input a number 0-15 using SW[3:0] (where SW3 is most significant bit, SW0 is least significant bit)
        ii. Use KEY3 to enter the value and move on to the next digit
        iii. Use KEY2 to remove the last entered value
        iv. Use KEY1 to clear all input
4. Read results from FPGA screen
    a. Original message is displayed on the top line
    b. Encrypted / decrypted result is displayed on the bottom line, once all 16 digits are entered
5. (Optional) Verify process with Java DES implementation.
    a. This file is included in the Github Repository

Citations:

1. The DES Algorithm Illustrated, by J. Orlin Grabbe
   http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

2. Youtube Demonstration
   https://youtu.be/E12OgBh-Sj8

3. Altera DE2 Project lcdlab1, by John Loomis
   http://www.johnloomis.org/digitallab/lcdlab/lcdlab1/lcdlab1.html

4. Keyboard Driver (keyboard.v and PS2_Controller.v) by Alejandro Cabrerizo and Will Zeurcher
   https://github.com/alecamaracm/ECE287Project/tree/master/VerilogExample