Jason Dykstra

# Problem 1 - DES

### Part a - Extend input to 48 bits

Given input: 01100110000111111001000101001 10

Given expansion table:

DES Expansion Table

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

(Bit 32 goes in position 0, bit 1 goes in position 1, etc.)

Expanded input: 010110100110100001011111111001010001010100001101

### Part b - Add (XOR) given round key to the expanded input

Given round key: 000010011100110001100010110111101011100000001111
XOR with expanded input from part a

Result: 010100111010010000111101001110111010110100000010

### Part c - Using 8 S-boxes, find 32 bit output of substitution step

Use result from part b to create 8 S-box inputs simply by separating every 8 bits

S-box inputs: 010100, 111010, 010000, 111101, 001110, 111010, 110100, 000010

Used this table to find 4 digit output of each S-box (image from wikipedia)

| $S_5$ | | Middle 4 bits of input | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0000** | **0001** | **0010** | **0011** | **0100** | **0101** | **0110** | **0111** | **1000** | **1001** | **1010** | **1011** | **1100** | **1101** | **1110** | **1111** |
| **Outer bits** | **00** | 0010 | 1100 | 0100 | 0001 | 0111 | 1010 | 1011 | 0110 | 1000 | 0101 | 0011 | 1111 | 1101 | 0000 | 1110 | 1001 |
| | **01** | 1110 | 1011 | 0010 | 1100 | 0100 | 0111 | 1101 | 0001 | 0101 | 0000 | 1111 | 1010 | 0011 | 1001 | 1000 | 0110 |
| | **10** | 0100 | 0010 | 0001 | 1011 | 1010 | 1101 | 0111 | 1000 | 1111 | 1001 | 1100 | 0101 | 0110 | 0011 | 0000 | 1110 |
| | **11** | 1011 | 1000 | 1100 | 0111 | 0001 | 1110 | 0010 | 1101 | 0110 | 1111 | 0000 | 1001 | 1010 | 0100 | 0101 | 0011 |

Output: 00110011100001010110001111001100

### Part d - Permute the S-box output using the permutation table

Given permutation table:

Permutation table

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

(Bit 16 goes in position 0, bit 7 goes in position 1, etc.)

Result: 11001000001101000111001110100011

All calculations/steps for this part can be found in this spreadsheet, if you're curious :)
https://docs.google.com/spreadsheets/d/12JRjCk4_APuoPmlZnTe8mhkawVXdUi4DTXcou0qDHLI/edit?usp=sharing

## Problem 2 - AES

### Part a - Convert the given 128 bit input to hexadecimal form

Given input:
01010110111000100001100110110010010001001011001111011011010000111000000100011110100111010011101010011111010000101111100110100111 1

Input in hexadecimal form: 56E219B244B3DB43811E9D3A9E85F34F
(With spaces: 56 E2 19 B2 44 B3 DB 43 81 1E 9D 3A 9E 85 F3 4F)

### Part b - Write the input in a state diagram (4x4 matrix)

Simply write out the hexadecimal representation in a 4x4 matrix, filling from top to bottom, left to right

State diagram:

| | | | |
|---|---|---|---|
| 56 | 44 | 81 | 9E |
| E2 | B3 | 1E | 85 |
| 19 | DB | 9D | F3 |
| B2 | 43 | 3A | 4F |

### Part c - Apply SubBytes step: use AES S-box to substitute the input

Given AES S-box table:

AES S-box Table

| | | y | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| x | 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| | 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| | 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| | 3 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| | 4 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| | 5 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| | 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| | 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| | 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| | 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| | a | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| | b | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| | c | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| | d | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| | e | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| | f | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

Result:

| | | | |
|---|---|---|---|
| B1 | 1B | 0C | 0B |
| 98 | 6D | 72 | 97 |
| D4 | B9 | 5E | 0D |

| | | | |
|---|---|---|---|
| 37 | 1A | 80 | 84 |

### Part d - Apply shift rows step

Shift first row left by 0 indexes
Shift second row left by 1 indexes
Shift third row left by 2 indexes
Shift fourth row left by 3 indexes
(With wrapping)

Result:

| | | | |
|---|---|---|---|
| B1 | 1B | 0C | 0B |
| 6D | 72 | 97 | 98 |
| 5E | 0D | D4 | B9 |
| 84 | 37 | 1A | 80 |

### Part e - Apply MixColumns Step

Using polynomial: $P(x) = x^8 + x^4 + x^3 + x + 1$

Transformation matrix used:

| | | | |
|---|---|---|---|
| 02 | 03 | 01 | 01 |
| 01 | 02 | 03 | 01 |
| 01 | 01 | 02 | 03 |
| 03 | 01 | 01 | 02 |

Essentially, Multiply transformation matrix above with the result from part d.

**Important:** Operations are as follows
02 • 01111111 = 11111110 (Shift by one)

- If bit shifted off is a 1, then XOR result with 1B (00011011)

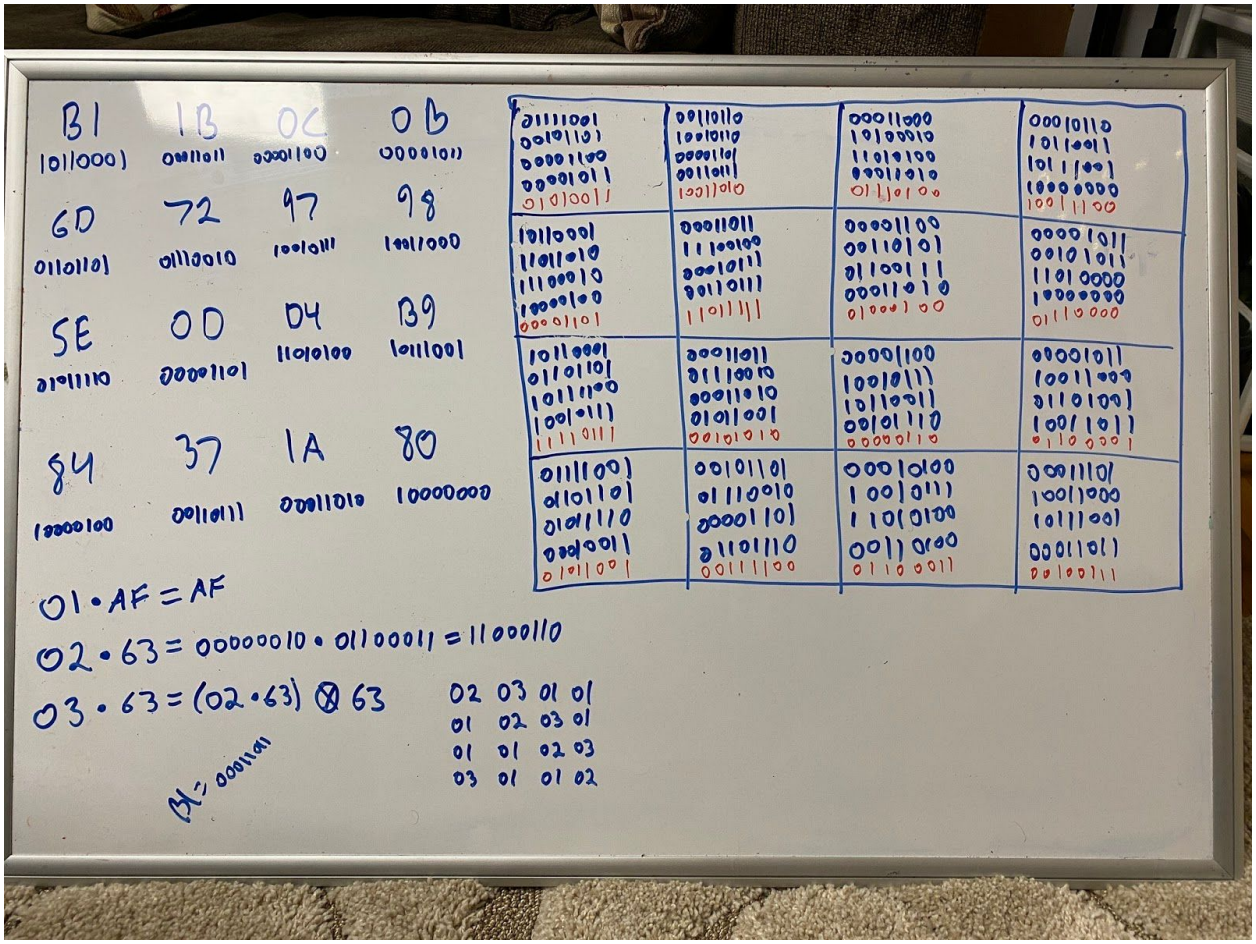
03 • 01111111 = (02 • 01111111) ⊗ 01111111 (perform 02 operation then XOR with original self)
01 • 01111111 = 0111111 (No modification)

Converted answer from part d to binary:

| | | | |
|---|---|---|---|
| 00010100 | 10011010 | 01110100 | 10011100 |
| 00001101 | 11011111 | 01000100 | 01110000 |
| 11110111 | 00101010 | 00000110 | 01100001 |

| | | | |
|---|---|---|---|
| 11101000 | 00111100 | 01100011 | 00100111 |

Picture of work:




(Note: cells 1 and 4 are incorrect, I fixed this after I took the picture)

Answer in hexadecimal:

| | | | |
|---|---|---|---|
| 14 | 9A | 74 | 9C |
| 0D | DF | 44 | 70 |
| F7 | 2A | 06 | 61 |
| E8 | 3C | 63 | 27 |

### Part f - add round key

Input (answer from part e in binary):
00010100000011011111011111101000100110101101111100101010001111000111010001000100000001100110001110011100011100000110000100100111

Given round key:
00110100000010011010011011010110011101101001001100101000010000111101010100000100110010001100110111110001101101010111001001110010

Answer (XOR of both):
00100000000001000101000100111110111011000100110000000010011111111101000010100000011001110101011100110110111000101000100110101010101

(Work for problems 1 and 2 can be found on this spreadsheet: https://docs.google.com/spreadsheets/d/12JRjCk4_APuoPmlZnTe8mhkawVXdUi4DTXcou0qDHLI/edit?usp=sharing)

## Problem 3 Modular Arithmetic

### Part a

1. 19
2. 17
3. 6
4. 2 (assuming the operation is: (15*29 + 11*15)mod23)

### Part b

1. 15
2. 7
3. 15
4. No modular inverse

### Part c

All numbers that are divisible by 2 and 3 are elements of modulo 216 with no multiplicative inverse