Maurizio Altamura
Cameron Lohr

CIS 3362 Homework #4: Hill Cipher, Bitwise Operators, DES, AES
Due: Check WebCourses for the due date.

Directions: To be done in pairs. If you can't find someone to work with, you must submit individually. When you submit, please CLEARLY mark both group members on each document you submit.

1) Consider a cipher that uses a 16 bit key and 16 bit blocks. Let A and B both be permutations matrices used in the cipher, assuming that A and B are expressed in a similar manner to how IP is expressed in DES. Let C be a matrix that represents the equivalent permutation to applying A, followed by applying B. (Thus, C(x) = B(A(x)), where x is a 16 bit input.) Determine C given the matrices A and B below:

$$A = \begin{bmatrix} 8 & 4 & 2 & 1 \\ 12 & 6 & 5 & 3 \\ 16 & 13 & 11 & 7 \\ 15 & 14 & 10 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 7 & 11 & 15 \\ 2 & 6 & 10 & 14 \\ 4 & 8 & 12 & 16 \\ 1 & 5 & 9 & 13 \end{bmatrix}$$

As we're applying B to A, we're going to be placing the bit in the first position with the designated bit from A by B. IE; B states that the 3rd entry of A should be first, so C(1) = 2. The second number should be the 7th position in A, so C(2) = 5. We continue this pattern down the entire B array until we have found the appropriate output, which results as:

$$C = \begin{matrix} 2 & 5 & 11 & 10 \\ 4 & 6 & 13 & 14 \\ 1 & 3 & 7 & 9 \\ 8 & 12 & 16 & 15 \end{matrix}$$

2) Imagine a DES-like cipher with a block size of 16 with the following IP matrix:

$$\begin{pmatrix} 10 & 13 & 16 & 9 \\ 2 & 5 & 15 & 7 \\ 3 & 12 & 11 & 14 \\ 4 & 8 & 1 & 6 \end{pmatrix}$$

What is the corresponding IP^-1 matrix?

If order to achieve an opposite IP matrix, we need to again use the provided matrix as a reference to where the values should be. This time, we use the position of the given matrix to place the values for the inverse. So, F(1) = 10, so N(10) = 1. This means that the tenth position of the new array will be 1. Looking at the second value, it is 13. This means that the 13th value in the new array will be 2. Continuing this pattern for the whole array, we get the following output:

15	5	9	13
6	16	8	12
4	1	11	10
2	12	7	3

3) If the input into all 8 S-boxes in DES is 1245789ABCDF, what is the output? Please express your output in 8 hexadecimal characters.

Before we begin using S-Boxes for this value, we need to convert all of the values back into Binary. This will provide us with the following output:

0001 | 0010 | 0100 | 0101 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1111

We'll then group the values into blocks of 6 to simulate the S-boxes.

000100 | 100100 | 010101 | 111000 | 100110 | 101011 | 110011 | 011111

We then concatenate the first and last bit of each block to find the row in the proper DES s-box table. In this first case, it would be [00] which is row 0. We then use the middle four bits, [0010] to determine the column, which appears to be 2. We then use the DES s-box tables for position 0 (As this is the first s-box we have made), which results in the value 13. We then convert 13 to binary once again, which leaves us with [1101]. We perform this again with the second s-box, with the row being [10], or 2. The column then becomes [0010], or 2 again. We then search the first s-box for the appropriate value, which is 7, which in binary is [0111]. We can perform this with each box to get the following:

1101 | 0111 | 0101 | 0101 | 1011 | 0101 | 0101 | 0010

Converting all of these Binary values back to Hexadecimal brings us to the following output:

D755B552

4) The first part of the function F in a round of DES expands the 32-bit input (from the right half of the previous round) to 48 bits. If this input, in HEX to the function F is 59E6BA91, what is the output of the expansion matrix. Express your answer as 12 hexadecimal characters.

Firstly, we must concert our hex values into binary:

0101 | 1001 | 1110 | 0110 | 1011 | 1010 | 1001 | 0001

Then we expand our binary values by grabbing the last digit from the block prior to the current one, and the first digit from the next block. Ie; the first block becomes 101011 as we take the previous number (1) from the 8th block and and a (1) from the second block as well. If we perform this on the entire system, we get the following:

101011 | 110011 | 111100 | 001101 | 010111 | 110101 | 010010 | 100010

After performing this operation, we should have the intended 48 bits. We may then convert each block in pairs of fours like so:

1010 | 1111 | 0011 | 1111 | 0000 | 1101 | 0101 | 1111 | 0101 | 0100 | 1010 | 0010

Which becomes…

AF3F0D5F54A2

5) In the specification of DES, the key is represented as 64 bits, of which some are parity bits. Label all the bits (including parity bits) as k_1, k_2, ..., k_64. If you knew the values of k_1 through k_24,but had to perform a brute force search through the other bits of the key, how long, in the worst case, would it take you to find the key, given that you can search through 2^20 keys in one second? Please express your answer in hours, minutes and seconds.

We know the bits K1 through K24, and the parity bits left are K32, K40, K48, K56, K64 (5 bits). So we're trying to find the remaining keys of 64 - 24 - 5, which is 35. To find our time then, we would multiply the number of keys by the time it takes us to search the values. We can divide by minutes, seconds, and even hours to find the appropriate measurement we would like.

Keys = 2^35 keys * 1sec/2^20 keys = 32,768 seconds.
32768 seconds * 1minute/60seconds = 546.3 minutes
546.3 * 1hour/60minutes = 9.1 hours

6) Let the input to the MixCols (during AES encryption) be $\begin{bmatrix} A0 & 74 & 65 & 96 \\ 2B & 8D & 2E & E3 \\ 99 & 1F & C8 & 87 \\ C5 & E5 & F7 & BB \end{bmatrix}$.

What's the output in row 3 col 4? (The matrix by which to "multiply" is $\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$.

So matrix multiplication of 4x4 * 4x4 = 4x4
For row 3 column 4, the equation is (96*01)+(E3*01)+(87*02)+(BB*03)

For multiplying by 1, it is itself.
For multiplying by 2, you left shift by 1
    If the highest bit had been 1, you xor with 1B
    If the highest bit had been 0, you don't do anything.
For multiplying by 3, you can first do the multiplication by 2, and then xor with the original value.

So converting all those hex values to binary and then applying the appropriate rule to them to multiply, and then xor all the parenthesis (adding), we get an answer.

(10010110) + (11100011) + (10011100) + (11010110)

(01110101) + (01001010)

00111111= 3F

7) In the key expansion algorithm of AES, if w[34] = BB3A7920 and w[31] = C659D034, what is w[35]?

Since 35 is not a multiple of 4, a simple xor is used on w[i-1] and w[i-4].

10111011001110100111100100100000+11000110010110011101000000110100

1111101011000111010100100010100 = 7D63A914

8) Consider the process of AES Key Expansion. Imagine that we have:

w[28] = B5 13 2F 76 (in hex)
w[31] = F4 9A 0D 8C (in hex)

Calculate w[32], showing each of the following intermediate results: RotWord(temp), SubWord(RotWord(temp)), Rcon[i/4], and the result of the XOR with Rcon[i/4].

RotWord	SubWord	Rcon[i/4]	XOR	FinalResult
9A 0D 8C F4	B8 D7 64 BF	80 00 00 00	38D764BF	8DC44BC9

For rotword, we take the first byte of w[i-1] and move it to the least significant space in that word.

For subword, we take rotword and replace each byte with the byte in the aes s box, using the left four bits for row and right 4 bits for column.

Rcon[i/4] = rcon[32/4] = rcon[8]

XORis the xor of rcon and subword

Final result is the xor of XOR and w[i-4]

9) Without examining all entries in the 16 round key schedule of DES, determine whether or not each number (which represents a bit location in the original key in each of the 16 boxes labeled "Round 1" through "Round 16") appears the exact same number of times collectively in the 16 boxes. (As an example, 10 appears in round except rounds 4, 12 and 14, so it appears 13 times.) Give proof of your answer.

Each box has 48 values in it, and there are 16 boxes, so a total of 16*48=768 available value spots in total. Each value has a 1/56 possibility. So 768/56 = 96/7. Thus, it does not divide evenly into the boxes, meaning there will be left over numbers, and not all the numbers will appear the same number of times in all 16 boxes.

10) Using the code, AES.java, posted in the course examples, write a program (preferably in Java, but C, C++ and Python will be accepted as well), that prints out a 256 x 256 chart which has the results of every possible byte multiplication in the AES field.Your program should output 256 rows, where the ith row stores the products with byte value i-127. Your output should express each value as 2 hex chars followed by a space. (So, each row should have 256 x 3 = 768 characters on it, followed by a new line character.) The hex letters should be lower case and leading 0s should be printed. In Java, we can accomplish this as follows:

```
System.out.printf("%02x ", byteToBePrinted);
```

Here is the beginning of the output of the few lines:

```
9b 03 82 28 a9 31 b0 7e ff 67
03 9e 1c bf 3d a0 22 fd 7f e2
82 1c 9f 3b b8 26 a5 75 f6 68
28 bf 3b 8a 0e 99 1d e0 64 f3
```

So, what we needed to do is create a 256x256 chart, which is a double for loop, and then call the function multiply to multiply the rows and columns (-127 to 127 = i-127 = 0-255). But we needed to call convert on the row and column numbers in order to convert it to a byte first and then multiply them together and output a hex number.

Attached code.