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:

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

**Solution**

Matrix B says to grab the 16th bit of output from matrix A and place it as the first bit. The 16th bit of output from matrix A was the 8th bit of output from the original input. This means that the first entry (row 1, col 1) of matrix C is 8.

The 13th bit of the output from matrix A is the 2nd bit of the original input. Thus, the entry in row 1 col 2 is 2. The 10th bit of output from matrix A is the 16th bit from the original input. Thus, the entry in row 1 col 3 is 16.

Continuing this work, the matrix C = \[
\begin{bmatrix}
8 & 2 & 16 & 11 \\
6 & 9 & 3 & 13 \\
7 & 10 & 4 & 15 \\
5 & 1 & 14 & 12
\end{bmatrix}
\]

Note: Solving the problem backwards yields C = \[
\begin{bmatrix}
10 & 1 & 9 & 2 \\
14 & 8 & 4 & 16 \\
6 & 3 & 11 & 15 \\
13 & 5 & 7 & 12
\end{bmatrix}
\]
2) Imagine a DES-like cipher with a block size of 16 with the following IP matrix:

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

What is the corresponding IP\(^{-1}\) matrix?

**Solution**

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

To solve this, note that the first matrix defines a function with \(f(1) = 6, f(2) = 13, f(3) = 7, ..., f(16) = 10\). It follows that \(f^{-1}(6) = 1, f^{-1}(13)=2, f^{-1}(7) = 3, ..., f^{-1}(10) = 16\). To implement this in IP\(^{-1}\), we place 1 in the sixth slot, 2 in the 13th slot, and so forth.

3) If the input into all 8 S-boxes in DES is 8df63098e724, what is the output? Please express your output in 8 hexadecimal characters.

**Solution**

To use the S-boxes, you need to first convert the input into eight 6-bit blocks; to get these, you must convert the given input into binary (I have added bars to illustrate the 6-bit blocks):

\((100011|011111|011000|110000|100110|001110|011100|100100)_2\). Now for each block, concatenate the 1\(^{st}\) and 6\(^{th}\) bits to get a 2-bit row value, and then concatenate the remaining middle bits (2\(^{nd}\) through 4\(^{th}\)) to get a 4-bit column value. For the first block, this will be row = (11)\(_2\) = 3 and column = (0001)\(_2\) = 1. Now use these values to look up a corresponding S-box output (with \(S_1\) being used for block 1, \(S_2\) being used for block 2, etc.) for each block. To find the output of \(S_1(b_1)\), determine the value within the S-box located at row 3, column 1 (note that row and column values are zero-based). This value is \(12 = (1100)_2\), and in binary it represents the first 4 bits of final 32-bit output: \((1100|0101|1011|1111|1011|1000|0110|0100)_2\). Now, convert this value to hexadecimal for the final result:

**C5BFBB864**
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 BF8293E6, what is the output of the expansion matrix. Express your answer as 12 hexadecimal characters.

**Solution**
Before using the expansion matrix (or performing the expansion algorithm), you must convert the given input into binary: $(1011|1111|1000|0010|1001|0011|1110|0110)_2$. From here, you can refer to the expansion table where each value within the table denotes the index of an input bit and where it lies within the output (e.g. the 32nd bit of the input is the 1st bit of the output, the 1st bit of the input is the 2nd bit of the output, and so on). Alternatively, you can use the algorithm that the table is based upon which might be quicker to use: simply divide the input into eight 4-bit blocks, and append the rightmost bit of the previous block to the leftmost bit of the current block and the leftmost bit of the following block to the rightmost bit of the current block. (For edge cases, simply wrap the first / last bit as necessary).

For example, to expand block 1 $(1011)_2$ of the given input, take the rightmost bit of the previous block (block 8 due to an edge case) - $(0110)_2$ and append it to the leftmost bit of block 1: $(0|1011)_2$. Now, take the leftmost bit of block 2 $(1111)_2$ and append it to the rightmost bit of block 1: $(0|1011|1)_2$. Do this for each block and you will receive a 48-bit binary result: $(010111|111111|110000|000101|010010|100111|111100|001101)_2$. Convert this to hexadecimal for the desired output:

5FFC054A7F0D

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 $k_{16}$, 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 days, rounded to the nearest day.

**Solution**
Known bits: $k_1, k_2, ..., k_{16}$
Parity bits: $k_{24}, k_{32}, k_{40}, k_{48}, k_{56}, k_{64}$
Unknown bits: $k_{17}, ..., k_{23}, k_{25}, ..., k_{31}, k_{33}, ..., k_{39}, k_{41}, ..., k_{47}, k_{49}, ..., k_{55}, k_{57}, ..., k_{63}$

$|\text{Key space}| = 2^{|\text{Unknown Bits}|} = 2^{64 - (|\text{Known bits| + Parity Bits})} = 2^{64 - (16 + 6)} = 2^{42}$

Time in days $= (2^{42} \text{ keys}) \times \left(\frac{1 \text{ sec}}{2^{20} \text{ keys}}\right) \times \left(\frac{1 \text{ min}}{60 \text{ sec}}\right) \times \left(\frac{1 \text{ hr}}{60 \text{ min}}\right) \times \left(\frac{1 \text{ day}}{24 \text{ hr}}\right)$

$= 48,545 \text{ days}$

$\approx 49 \text{ days}$

To find the solution to this question, you need to determine the size of the key space that you need to search ($2^{|\text{Unknown Bits}|}$) and then use simple conversions based on your search rate ($2^{20}$ keys per second) to calculate the time in days that it would take to brute-force this many keys.
6) Let the input to the MixCols (during AES encryption) be \[
\begin{bmatrix}
A_0 & 74 & 65 & 96 \\
2B & BD & 2E & E3 \\
99 & 1F & C8 & 37 \\
C5 & E5 & F7 & BB
\end{bmatrix}.
\]

What’s the output in row 4 col 1? (The matrix by which to “multiply” is \[
\begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
03 & 01 & 01 & 02
\end{bmatrix}.
\]

**Solution**

Before multiplying anything, you must first determine which rows and columns of the above matrices need to be multiplied together to get the output in row 4 column 1 of the resultant matrix. Based on the definition of matrix multiplication, we find that the target value is \((AB)_{ij} = \sum_k A_{ik} B_{kj}\) where A is the second matrix, B is the first matrix, AB is the resultant matrix, i is the row number, j is the column number, and m (in this case) is the length / width of matrices. Fitting the formula for the given problem gives us \((AB)_{41} = (03 \times A0) + (01 \times 2B) + (01 \times 99) + (02 \times C5)\). Now, all that is left is the multiplication and addition (XOR) of the terms based on the rules of multiplication in the field GF(2^8) covered in the chapter 7 lecture notes:

\[
03 \times A0 = (02 \times A0) \oplus (01 \times A0) \\
= (5B \oplus A0) \\
= F1
\]

\[
02 \times A0 = (A0 \ll 1) \oplus 1B \\
= 1|0100|0000 \\
\oplus 0001|1011 \\
\hline
0101|1011 = 5B
\]

\[
01 \times A0 = A0
\]

\[
5B \oplus A0 = 0101|1011 \\
\oplus 1010|0000 \\
\hline
1111|1011 = FB
\]

\[
01 \times 2B = 2B \\
= 0010|1011
\]

\[
01 \times 99 = 99 \\
= 1001|1001
\]

\[
02 \times C5 = (C5 \ll 1) \oplus 1B \\
= 1|1000|1010 \\
\oplus 0001|1011 \\
\hline
1001|0001 = 91
\]
(AB)_{41} = (03 \times A0) + (01 \times 2B) + (01 \times 99) + (02 \times C5)
   = FB \oplus 2B \oplus 99 \oplus 91
   = 1111|1011
   0010|1011
   1001|1001
   \oplus 1001|0001
   \hline
   1101|1000 = \textbf{D8}

7) In the key expansion algorithm of AES, if w[26] = 8EFA5329 and w[23] = 7EE826D3, what is w[27]?

**Solution**

Because (27 mod 4) \neq 0 (thus w[27] is not a new round key), you need only calculate

w[i] = w[i - 4] \oplus w[i - 1], and more specifically:

w[27] = w[23] \oplus w[26]
   = 7EE826D3 \oplus 8EFA5329
   = 0111|1110|1110|1000|0010|0110|1101|0011
   \oplus 1000|1110|1111|1010|0101|0011|0010|1001
   \hline
   1111|0000|0001|0010|0111|0101|1111|1010 = \textbf{F0 12 75 FA}
8) Consider the process of AES Key Expansion. Imagine that we have:

\[ w[36] = 3A \text{ 74 E5 8D (in hex)} \]
\[ w[39] = 8F \text{ 17 60 C2 (in hex)} \]

Calculate \( w[40] \), 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].

**Solution**

<table>
<thead>
<tr>
<th>RotWord</th>
<th>SubWord</th>
<th>Rcon[i/4]</th>
<th>XOR</th>
<th>FinalResult</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 60 C2 8F</td>
<td>F0 D0 25 73</td>
<td>36 00 00 00</td>
<td>C6 D0 25 73</td>
<td>FC A4 C0 FE</td>
</tr>
</tbody>
</table>

Note: unlike question 7, the key expansion this question is used to find \( w[40] \) which is a new round key because \((40)\text{mod}4 = 0\). This means we need to go through all of the steps listed on the chart above.

**RotWord**
Remove the first byte of \( w[i - 1] \) and append it to the end of \( w[i - 1] \) before sending this new value to SubWord: RotWord(w[39]) = 17 60 C2 8F.

**SubWord**
Using the result of the previous step, replace each byte with a value given in the AES S-box. To look up the correct value, for each byte: the left 4 bits represent the row, and the right 4 bits represent the column. Find the corresponding byte value (given in hexadecimal) and replace each byte accordingly: SubWord(17 60 C2 8F) = F0 D0 25 73.

**Rcon[i/4]**
Use the hexadecimal value given at index \( i/4 \) in the Rcon table to XOR with the value found in the previous step. Rcon[40/4] = Rcon[10] = 36 00 00 00.

**XOR**
Simply convert the results of the previous two steps to binary and XOR them:
\[
F0D02573 \oplus 36000000 = 1111|0000|1110|0000|0010|0101|0111|0011
\oplus 0011|0110|0000|0000|0000|0000|0000|0000
\hline
1100|0110|1101|0000|0101|0101|0111|0011 = C6 D0 25 73
\]

**Final Result**
Now, simply perform another XOR on the previous result and \( w[i/4] \) to determine the final answer:
\[
C6D02573 \oplus 3A74E58D = 1100|0110|1111|0000|0010|0101|0111|0011
\oplus 0011|1010|0011|0100|1110|0101|1000|1101
\hline
1111|1100|1010|0100|1100|0000|1111|1110 = FC A4 C0 FE
\]
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.

**Solution**
Each number does not appear the same number of times in the 16 boxes. To see this, note that there are 48 values in each box, so there are a total of 16 x 48 values. Each value is one of 56 possible ones, thus the average number of times each value appears is \( \frac{16 \times 48}{56} = \frac{16 \times 6}{7} = \frac{96}{7} \). Since this number isn’t an integer, it’s impossible that each of the values appears exactly this many times. Instead, it must be the case that some values appear more than this number of times while other values appear less than this. (This number is in between 13 and 14. So 10 appears slightly less often than the average number.)

10) Consider an AES plaintext of

\[
\begin{bmatrix}
01 & 89 & FE & 76 \\
23 & AB & DC & 54 \\
45 & CD & BA & 32 \\
67 & EF & 98 & 10
\end{bmatrix}
\]

with a key of 128 1s. Show the state matrix after the shift rows step in Round 1.

**Solution**
First we XOR with the key, since it’s all 1’s, we "reflect" each byte:

\[
\begin{bmatrix}
FE & 76 & 01 & 89 \\
DC & 54 & 23 & AB \\
BA & 32 & 45 & CD \\
98 & 10 & 67 & EF
\end{bmatrix}
\]

Then, substitute each byte using the s-box to get:

\[
\begin{bmatrix}
BB & 38 & 7C & A7 \\
86 & 20 & 26 & 62 \\
F4 & 23 & 6E & BD \\
46 & CA & 85 & DF
\end{bmatrix}
\]

Finally, for each row i, perform a circular left shift of i bytes (with i and row number starting at zero):

\[
\begin{bmatrix}
BB & 38 & 7C & A7 \\
20 & 26 & 62 & 86 \\
6E & BD & F4 & 23 \\
DF & 46 & CA & 85
\end{bmatrix}
\]