1. Mix Cols Review
2. AES Key Schedule
3. Helper Mix Cols Program
4. Helper DES S-box Program
5. AES code

Mix Cols
---------

<table>
<thead>
<tr>
<th>02</th>
<th>03</th>
<th>01</th>
<th>01</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>03</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>02</td>
<td>03</td>
</tr>
<tr>
<td>03</td>
<td>01</td>
<td>01</td>
<td>02</td>
</tr>
</tbody>
</table>

x

<table>
<thead>
<tr>
<th>63</th>
<th>7c</th>
<th>77</th>
<th>93</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>3f</td>
<td>cd</td>
<td>26</td>
</tr>
<tr>
<td>f1</td>
<td>1a</td>
<td>0c</td>
<td>13</td>
</tr>
<tr>
<td>16</td>
<td>4b</td>
<td>c6</td>
<td>d2</td>
</tr>
</tbody>
</table>

Note: first matrix is fixed for the algorithm.
Second matrix is the state matrix right before Mix Cols.

Entry in row 2, column 1:

\[01 \times 63 + 02 \times 36 + 03 \times f1 + 01 \times 16\]
\[63 + 6c + 08 + 16\]

Rules for multiplication in the field:
1. mult 2 is a left shift
2. If there is an overflow bit, that bit is equivalent to xoring with 0001 1011, from the AES polynomial.
3. All other multiplications can be created as a combination of these rules and addition, which is XOR in the field.
02 x 36 = 0011 0110 left shifted by 1 bit
 0110 1100 = 6c

03 x f1 = 02 x f1 +
 01 x f1

02 x f1 = 1111 0001 left shifted by 1
  = 1 1110 0010 (overflow
  = 1110 0010
  + 1 1011
  ----------
  1111 1001

03 x f1 = 1111 1001 (02 x f1)
  + 1111 0001 (01 x f1)
  ----------
  0000 1000 = 08

Final answer:

63 + 6c + 08 + 16 use the XOR chart for HEX

6 + 6 + 0 + 1 = 1 (xor)
3 + c + 8 + 6 = f + e = 1

Final Answer is 11.

**AES Key Schedule**
bytes are 8 bits, words are 32 bits, so 4 bytes = word

KeyExpansion(byte key[16], word w[44]) {

    word temp;
    for (i=0; i<4; i++) w[i] = (key[4*i],key[4*i+1],key[4*i+2],key[4*i+3]);

    for (i=4; i<44; i++) {
        temp = w[i-1];
        if (i%4 == 0) // Beginning of the new round key
            temp = SubWord(RotWord(temp)) XOR Rcon[i/4]
\[ w[i] = w[i-4] \text{ XOR } \text{temp}; \]

Round 0 key is the original key.
For round 1 to round 10, the keys are:
- \( w[4] \ldots w[7] \) R1
- \( w[4i] \ldots w[4i+3] \) Round i key

Unless it's the beginning of a new round key, all we do to generate the next 32 bits, is XOR the last 32 bits with the 32 bits from 4 times ago.

But, when we begin a new round, we want to mess stuff up... so here is what we do:

RotWord is a cyclic left rotation by one byte.

i=36 in this example...
Let's say that \( w[35] = \text{7f8d292f} \) (in hex)
\( \text{RotWord}(w[35]) = \text{8d292f7f} \)

Next step is to do a SubBytes on the buffer:

\( \text{SubBytes(8d)} = \text{5d} \) (from S box)
\( \text{SubBytes(29)} = \text{a5} \) (from S box)
\( \text{SubBytes(2f)} = \text{15} \) (from S box)
\( \text{SubBytes(7f)} = \text{d2} \)

So new buffer is \( \text{5da515d2} \), after subbytes.

XOR with \( \text{Rcon[36/4]} = \text{Rcon[9]} \).

Here are the Rcon values in hex:

<table>
<thead>
<tr>
<th>I</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rcon[i]</td>
<td>01000000</td>
<td>02000000</td>
<td>04000000</td>
<td>08000000</td>
<td>10000000</td>
<td>20000000</td>
<td>40000000</td>
<td>80000000</td>
<td>1B000000</td>
<td>36000000</td>
</tr>
</tbody>
</table>

Our example:
5da515d2 XOR 1b000000 (since round 9)

46a515d2, this is our new buffer (for temp)

Finally to get w[i], we have to XOR it with w[i-4], so I need to give you the value of w[32].

For this example w[32] = ead27321

4 6 a 5 1 5 d 2
e a d 2 7 3 2 1
-----------------
a c 7 7 6 6 f 3

Sbox program verification
--------------------------
Input: f3e = 1111 0011 1110

Sbox1(111100) = row 10 = 2, col = 1110 = 14

For Quiz
--------
Preprint all the necessary tables.
DES tables
AES tables
Bin->Hex chart