1 Problem 1

\[ X = 1001011010000001 \]  \hspace{1cm} (1)

We can calculate \( u_1 \) by XORing the input with the key.

\[ w_0 = 1001\ 0110\ 1000\ 0001 \]
\[ K_1 = 0011\ 1010\ 1001\ 0100 \]  \hspace{1cm} (2)
\[ u_1 = 1010\ 1100\ 0001\ 0101 \]

Now that we have \( u_1 \), we can find \( v_1 \) through permutation. If \( u_1 = 0xAC15 \), then \( v_1 = 0x654F \).

We can then find \( w_1 \) by permuting the bits.

\[ u_1 = 1010\ 1100\ 0001\ 0101 \]
\[ v_1 = 0110\ 0101\ 0100\ 1111 \]  \hspace{1cm} (3)
\[ w_1 = 0001\ 1111\ 1001\ 0101 \]

Therefore, \( w_1 = 0x1F95 \).
2 Problem 2

The piling up lemma formula makes the assumption that events are independent. Without this assumption, the formula could not be derived.

The piling up lemma derivation assumes that the probability that both $X_1$ and $X_2$ are either 0 or 1 is simply the product of each individual probability.

In essence,

$$\Pr[X_1 \oplus X_2 = 0] = \Pr[X_1 = 0]\Pr[X_2 = 0] + \Pr[X_1 = 1]\Pr[X_2 = 1] \quad (4)$$

This equation would not hold true if $X_1$ and $X_2$ were not independent of each other.
3 Problem 3

If $X_1 \oplus X_2$ and $X_2 \oplus X_3$ are independent, then the following must be true.

$$X_1 \oplus X_3 = (X_1 \oplus X_2) \oplus (X_2 \oplus X_3) \quad (5)$$

In terms of pilling-up lemma, we find the following.

$$2\epsilon_1 \epsilon_3 = 2(2\epsilon_1 \epsilon_2)(2\epsilon_2 \epsilon_3) \quad (6)$$

$$\frac{1}{4} \epsilon_1 \epsilon_3 = \epsilon_1 \epsilon_2^2 \epsilon_3 \quad (7)$$

If $\alpha = \epsilon_1 \epsilon_3$, then we obtain the following

$$\frac{1}{4} \alpha = \alpha \epsilon_2^2 \quad (8)$$

Here we can see that this holds true if and only if $\alpha = 0$, or if $\epsilon_2 = \pm \frac{1}{4}$.

Therefore, $X_1 \oplus X_2$ and $X_2 \oplus X_3$ are independent if either $\epsilon_1 = 0$, $\epsilon_3 = 0$, or $\epsilon_2 = \pm \frac{1}{4}$. 
4 Problem 4

4.1 Part A

The biases for the given S-box are as follows. The table was created using the C program found on the last two pages of this document.

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>12</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>12</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Program used for the table generated in Problem 4. Compiled on the CVL machines using gcc version 3.4.6.

#include <stdio.h>

static char permute(int val)
{
    static const char ptable[16] = {
        0x8, 0x4, 0x2, 0x1,
        0xC, 0x6, 0x3, 0xD,
        0xA, 0x5, 0xE, 0x7,
        0xF, 0xB, 0x9, 0x0
    };
    return ptable[val];
}

static char mask(char val, char m)
{
    return (val & 0xF) & m;
}

static char xorbits(char val)
{
    int res = 0;
    while (val != 0) {
        res ^= (val & 0x1);
        val = val >> 1;
    }
    return res;
}

static int calcbias(int a, int b)
{
    int i, sum = 0;
    int x, y;
    for (i = 0; i < 16; i++) {
        x = mask(i, a);
        y = mask(permute(i), b);
        if (xorbits(x) ^ xorbits(y) == 0)
            sum++;
    }
int main(int argc, char ** argv)
{
    int a, b, bias;
    for (a = 0; a < 16; a++) {
        for (b = 0; b < 16; b++) {
            bias = calcbias(a, b);
            printf("%2d ", bias);
        }
        printf("\n");
    }
    return 0;
}