Day 26 – Data Encryption Standard (DES)
Input: \( B = b_1 b_2 b_3 \ldots b_{64} \)

\[ \begin{array}{cccccccc}
58^{th} \text{ position} & \rightarrow & 56^{th} \text{ position} & \rightarrow & 42^{nd} \text{ pos.} & \ldots \\
1^\text{st} & 3^\text{rd} & 5^\text{th} & \ldots & \\
58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 \\
60 & 52 & 44 & 36 & 28 & 20 & 12 & 4 \\
62 & 54 & 46 & 38 & 30 & 22 & 14 & 6 \\
64 & 56 & 48 & 40 & 32 & 24 & 16 & 8 \\
57 & 49 & 41 & 33 & 25 & 17 & 9 & 1 \\
59 & 51 & 43 & 35 & 27 & 19 & 11 & 3 \\
61 & 53 & 45 & 37 & 29 & 21 & 13 & 5 \\
63 & 55 & 47 & 39 & 31 & 23 & 15 & 7 \\
\end{array} \]

**INITIAL PERMUTATION**

\[ \begin{array}{cccccccc}
1^\text{st} & 3^\text{rd} & 5^\text{th} & \ldots & \\
40 & 8 & 48 & 16 & 56 & 24 & 64 & 32 \\
39 & 7 & 47 & 15 & 55 & 23 & 63 & 31 \\
38 & 6 & 46 & 14 & 54 & 22 & 62 & 30 \\
37 & 5 & 45 & 13 & 53 & 21 & 61 & 29 \\
36 & 4 & 44 & 12 & 52 & 20 & 60 & 28 \\
35 & 3 & 43 & 11 & 51 & 19 & 59 & 27 \\
34 & 2 & 42 & 10 & 50 & 18 & 58 & 26 \\
33 & 1 & 41 & 9 & 49 & 17 & 57 & 25 \\
\end{array} \]

**INVERSE PERMUTATION**

58th bit after 16 stages will go to the 1st position in the ciphertext.
Split the string we obtain after the initial perm. into half $L$ and $R$.

"Add" to the left half the output $F(R_o, K_1)$ run $R_o$ through an F-box with the key $K_1$.

For general step $i$ (not last step)

$$\begin{align*}
L_{i+1} &= R_i \\
R_{i+1} &= L_i + F(R_i, K_{i+1})
\end{align*}$$
The last step is different.

\[ R_{16} = R_{15} \]

\[ L_{16} = L_{15} + F(R_{15}, K_{16}) \]

\[ R_{14} \]
16 STAGES

The F Function (Feistel Function)

1) Expand: 32 bits $\rightarrow$ 48 bits
2) Key padding
   - Generate a 48 bit key
   - "Add" this key to the expanded input
3) Compress: 48 bits $\rightarrow$ 32 bits
4) Mix-mash: Take output from step 3 and permute them under a perm.

"Add" is XOR operation (Exclusive OR)

\[
\begin{align*}
1 + 1 &= 0 \\
0 + 0 &= 0 \\
1 + 0 &= 1 \\
0 + 1 &= 1 \\
\end{align*}
\]

Addition under Mod 2.
Take 32-bit input
 Duplicate bits #1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29
 (omitting duplicates in the red box above)

32-bit → 48-bit

Rearrange the string according to the table above.

$S_{32}, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, \ldots, S_{22}, S_1$
\[ 1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13 \]

**Compression Schedule**

For input 1101:
- The row is 11-1.
- The column is 1101, 13.
- The output is 5 = 0101.

Example:
- Col 1111 = 15
- Row: 11 = 3
- Number on row 3, col 15 is 13
- 1101

- Take 48-bit string, break it into 8 groups of 6-bit substrings.
- Compress each 6-bit substring into a 4-bit using each of the $S_i$ table.

\[ S = s_1 s_2 s_3 s_4 s_5 s_6 \]

- $s_1 s_6$ in decimal form gives the row position of the $S$ table.
- $s_2 s_3 s_4 s_5$ in decimal form gives the col. position.

The compression of $S$ is the binary form for the corresponding entry in the table.
FURTHER MISH-MUSHING

To spread the 4-bit output of each S-box across different boxes in the next round.
The key in DES is a 56-bit string obtained by:

1. Randomize a 64-bit string
2. Discard 1 row of PC-1 table and the corresponding bits
3. Rearrange the remaining 56-bit according to PC-1 table.

**Permuted Choices**

Select 24-bits from the 28-bits $C_i$

Select 24-bits from the 28-bits $D_i$

Construct a round key $K_i$ which has 48-bits
KEY PROCESSING

64 bits
28 bits
28 bits
48 bits
28 bits
48 bits

concatenate then run through PC-2 to compress the 56-bit key into a 48-bit round key

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Number of Left Shifts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
</tr>
</tbody>
</table>

skip the entries to the left by 1 or 2 places.
\[ f \oplus g = r \text{ then to recover } f: \quad r \ominus g = f \]

To decrypt, run through the same machine in reverse order.