Section 4 Lecture 25 – DES Implementation - Solution

(i) Split the keyword into blocks of 7 with gaps between them

1001010 0011001 0111010 1010101 1101111 1101010 1010101 0101110

Then fill in the even parity bits, so the 64-bit keyword is

10010101001100110111010010101010110111101101010010101010010111 00

(ii) Probably easier to write the message with bit positions indicated:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1

25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
0	0	1	1	1	0	1	1	0	1	0	1	1	0	0	0	1	1	1	0	1	0	1	0

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
1	0	1	0	0	0	0	1	1	1	1	0	1	1	1	0

Now apply the permutation, writing down the bit in position 58, then in position 50, etc, which gives the permuted message

1011000000011010100000100100111011100010111011111011111110101001

The first half of this is 10110000000110101000001001001110

(iii) The blocks of 4 are 1011 0000 0001 1010 1000 0010 0100 1110

Follow the rules, so the first block of 6 is the bit "before" the first block – this means we need to wrap around to the end, so is 0, then write the block of 4 which is 1011, and then the bit "after" the block which is 0. This gives 010110 100000 000011 110101 010000 000100 001001 011101, or as a single block,
010110100000000001111010101000000010000010010111 01

(iv) Again, probably easier to write the keyword with the message positions included:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	1	1	1	0	1	0	0

25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
1	0	1	0	1	0	1	0	1	1	0	1	1	1	1	0	1	1	0	1	0	1	0	0

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
1	0	1	0	1	0	1	0	0	1	0	1	1	1	0	0

Then applying the first permutation (bit 57, then bit 49…) gives

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	1	1	1	1	0	0	1	1	0	1	1	0	1	0	0	0	1	0	0	1	1	1	0

25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
1	0	1	1	0	1	0	1	1	0	1	0	1	0	1	1	0	1	0	1	1	1	0	1

49	50	51	52	53	54	55	56
1	0	0	0	0	1	1	1

Then apply the second permutation (bit 14, then bit 17, …) to this to get the 48-bit block
101001110010101101001001000101110111111100110001

(v)

XOR'ing the two 48-bit blocks

010110100000000011110101010000000100001001011101
101001110010101101001001000101110111111100110001

together gives

111111010010101110111100010101110011110101101100

(vi) In blocks of 6 bits this is

111111 010010 101110 111100 010101 110011 110101 101100

Looking these up in the appropriate S-box in turn (so S_1 for the first block, then S_2 for the second block, and so on, this gives) 13 7 0 8 15 14 0 14, which in binary gives
1101 0111 0000 1000 1111 1110 0000 1110

Hence our 32-bit block is now 11010111000010001111111000001110

(vii) Applying the permutation (you'll probably write it with message positions included again) gives the final 32-bit half block as

01111001101001101100000011111010

(viii) Now repeat the process for the second half, which is

11100010111011111011111110101001

I won't give all the working (method as above, but the steps are given)

Applying (iii) gives the 48-bit block
111100000101011101011111110111111111110101010011

For (iv), the two permutations on the keyword are exactly the same.

For (v), the XOR of

111100000101011101011111110111111111110101010011
101001110010101101001001000101110111111100110001

gives the 48-bit block

010101110111110000010110110010001000001001100010

For (vi), the S-boxes on the blocks 010101 110111 110000 010110 110010 001000 001001 100010 give 12 12 11 5 9 9 4 11 which in binary is 1100 1100 1011 0101 1001 1001 0100 1011, and hence the 32-bit keyword is
11001100101101011001100101001011

Applying the permutation in (vii) we get 10111101100110101011100100010100 as our second 32 bit block

(ix) Join together the two 32-bit blocks (the second one first) to get the 64-bit block

1011110110011010101110010001010001111001101001101100000011111010

Remember at this point, in practice we would repeat this round a further 15 times so this is highly simplified!

(x) If the initial permutation is

{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}

then the inverse permutation is

{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}

(e.g. 40 is obtained since the 40^{th} digit of the initial permutation is bit 1 of the keyword, and so on)

(xi) Finally, applying this permutation to our 64-bit block

1011110110011010101110010001010001111001101001101100000011111010

gives us the final encrypted message as

1100010000110010011000011101011011010111111001101000101001111110