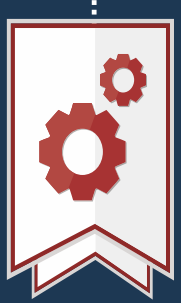

SIL765 - Assignment 1

Data Encryption Standard Implementation

By:
Yash Gupta (2013CS10302)
Ujjwal Sinha (2012CH70185)

DES - Introduction:

- DES is a block cipher, meaning that it operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size.
- It is based on Fiestel cipher, and has 16 rounds.
- Input block of 64 bits is divided into two blocks of 32 bits each, a left half block L and a right half R.
- Before the Fiestel rounds, the input block is permuted using a special initial permutation, which is inverted after the Fiestel rounds.

Project Overview

- We implement DES using C/C++
- All blocks are stored as unsigned integers (32-bit or 64-bit)
- There are 2 basic functions:
  encrypt(plain, key, outputs) and **decrypt(cipher, key, outputs)**
- Key schedule is implemented by
  key_round(i, CD) and inv_key_round(i, CD)
- Fiestel box is implemented by **fiestel_box(R,K)**
- To assist these functions, following are also used:
  initial_permutation(), inv_initial_permutation();
  expansion_box(), selection_box(), permutation_box();
  permutation_choice1_box(), permutation_choice2_box(), shift_boxes(), …

DES Encryption

- Here we take plain-text to be “FEED1337BEAD8787” and key to be “0E329232EA6D0D73” in hexadecimal format.
- Both plain text and the key is converted to binary format and we get a 64 bit block of plain text and key.
- Out of 64 bits of key, 8 bits of key are redundant parity bits.
- Initial permutation is applied to plain text.
- Permuted plaintext is split into L and R; key is used to initiate key schedule.
- Iterate over 16 rounds by invoking f-box, XOR’ing and swapping appropriately
- Perform final swap and invert initial permutation.
- Cipher-text = “1A4AE1F807D29195”

DES Encryption Output

```
plain: FEED1337BEAD8787
key:   0E329232EA6D0D73
00 - L: 031DFBEE  R: F33B33DD
01 - L: F33B33DD  R: 51D8987D  K: 36146478E1E1  CD: 0029617503EC2E3D
02 - L: 51D8987D  R: A9C71419  K: 40BD1176E8FD  CD: 0052C2EA07D85C7A
03 - L: A9C71419  R: 38255661  K: 45A473239DDB  CD: 004B0BA81F6171E9
04 - L: 38255661  R: F7F0DE37  K: E7C4828FB533  CD: 002C2EA05D85C7A7
05 - L: F7F0DE37  R: EC67ABB5  K: 7A83826F4F64  CD: 00B0BA8146171E9F
06 - L: EC67ABB5  R: EA49F160  K: 38901B58C9DE  CD: 00C2EA05285C7A7D
07 - L: EA49F160  R: D46E0D99  K: 25005EC5D49D  CD: 000BA814B171E9F6
08 - L: D46E0D99  R: 8FFCB3E7  K: 264894CB36E9  CD: 002EA052C5C7A7D8
09 - L: 8FFCB3E7  R: 72EBB23C  K: 54554179F633  CD: 005D40A58B8F4FB0
10 - L: 72EBB23C  R: C0AF3554  K: 43C9453F4C2E  CD: 007502961E3D3EC2
11 - L: C0AF3554  R: E849DBA6  K: 09E1878C79D6  CD: 00D40A5858F4FB0B
12 - L: E849DBA6  R: 3C314F65  K: 3105ABA5E2F5  CD: 0050296173D3EC2E
13 - L: 3C314F65  R: 85577A6D  K: F100A1F38EC3  CD: 0040A585DF4FB0B8
14 - L: 85577A6D  R: 5465A8F8  K: 918A949E871F  CD: 000296175D3EC2E3
15 - L: 5465A8F8  R: EC0C0B33  K: 1432961F77C4  CD: 000A585D44FB0B8F
16 - L: EC0C0B33  R: 2EE990D4  K: 606F044C3AE7  CD: 0014B0BA89F6171E
LR: 2EE990D4EC0C0B33
cipher: 1A4AE1F807D29195
```

DES Decryption

- Take cipher-text to be “1A4AE1F807D29195” and key to be “0E329232EA6D0D73”
- Apply initial permutation, split into two 32-bit blocks and swap.
- Iterate over 16-rounds in the reverse direction to invert the encryption
  As XOR is it’s own inverse, this is trivial
- For key schedule, reverse the shift schedule and right-shift instead of left.
  Works because total amount shifted equals 28 (for each 28-bit bock C and D).
- Finally obtain plain-text by combining L and R and applying inverse initial permutation.
- Plain text = “FEED1337BEAD8787”

DES Decryption Output

```
cipher: 1A4AE1F807D29195
key:    0E329232EA6D0D73
00 - L: EC0C0B33  R: 2EE990D4
01 - L: 5465A8F8  R: EC0C0B33  K: 606F044C3AE7  CD: 000A585D44FB0B8F
02 - L: 85577A6D  R: 5465A8F8  K: 1432961F77C4  CD: 000296175D3EC2E3
03 - L: 3C314F65  R: 85577A6D  K: 918A949E871F  CD: 0040A585DF4FB0B8
04 - L: E849DBA6  R: 3C314F65  K: F100A1F38EC3  CD: 0050296173D3EC2E
05 - L: C0AF3554  R: E849DBA6  K: 3105ABA5E2F5  CD: 00D40A5858F4FB0B
06 - L: 72EBB23C  R: C0AF3554  K: 09E1878C79D6  CD: 007502961E3D3EC2
07 - L: 8FFCB3E7  R: 72EBB23C  K: 43C9453F4C2E  CD: 005D40A58B8F4FB0
08 - L: D46E0D99  R: 8FFCB3E7  K: 54554179F633  CD: 002EA052C5C7A7D8
09 - L: EA49F160  R: D46E0D99  K: 264894CB36E9  CD: 000BA814B171E9F6
10 - L: EC67ABB5  R: EA49F160  K: 25005EC5D49D  CD: 00C2EA05285C7A7D
11 - L: F7F0DE37  R: EC67ABB5  K: 38901B58C9DE  CD: 00B0BA8146171E9F
12 - L: 38255661  R: F7F0DE37  K: 7A83826F4F64  CD: 002C2EA05D85C7A7
13 - L: A9C71419  R: 38255661  K: E7C4828FB533  CD: 004B0BA81F6171E9
14 - L: 51D8987D  R: A9C71419  K: 45A473239DDB  CD: 0052C2EA07D85C7A
15 - L: F33B33DD  R: 51D8987D  K: 40BD1176E8FD  CD: 0029617503EC2E3D
16 - L: 031DFBEE  R: F33B33DD  K: 36146478E1E1  CD: 0014B0BA89F6171E
LR: 031DFBEEF33B33DD
plain: FEED1337BEAD8787
```

DES Validation:

- To validate DES implementation: Output of the J^{th} encryption round should be identical to the output of the $(16-J)^{th}$ decryption round.
- In the encrypt() and decrypt() functions, store the intermediate outputs in an array.
- Finally assert that J^{th} and $(16-J)^{th}$ outputs of encrypt and decrypt respectively are equal.
- The program executes successfully with no assertion faliures, thus validating our implementation.

THANK YOU!