

CNS Lab Assignment 4: SDES

111703013 Akshay Deodhar

5th October 2020

Code

```
1
2
3  #include <sys/types.h>
4  #include <unistd.h>
5  #include <fcntl.h>
6  #include <errno.h>
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <stdint.h>
10
11  /* S-DES permutations */
12
13  typedef uint8_t u8;
14
15  typedef uint32_t u32;
16
17  static const u8 IPlen = 8;
18  static const u8 IP[] = {1, 5, 2, 0, 3, 7, 4, 6};
19
20  static const u8 invIPlen = 8;
21  static const u8 invIP[] = {3, 0, 2, 4, 6, 1, 7, 5};
22
23  static const u8 n_rounds = 2;
24
25  static const u8 P8len = 8;
26  static const u8 P8[] = {5, 2, 6, 3, 7, 4, 9, 8};
27
28  static const u8 P10len = 10;
29  static const u8 P10[] = {2, 4, 1, 6, 3, 9, 0, 8, 7, 5};
30
31  static const u8 EPlen = 8;
32  static const u8 EP[] = {3, 0, 1, 2, 1, 2, 3, 0};
33
34  /* rotate x, an input of m bits, n bits to the left */
35  /* note: the macro expects safe inputs (having the specified number of bits */
36  /* this might "seem" a rightshift- but in the "abstract" representation, the
37   * leftmost bit, is bit 0, which is actually the LSB */
38  u32 rotate_l(const u32 x, const u8 m, const u8 n) {
39      return ((1 << m) - 1) & ((x >> n) | (x << (m - n)));
40  }
41
42  /* append nl LSBs of l to nr LSBs of r, with l at the MSBs in result */
43  /* note: the macro expects safe inputs (having the specified number of bits */
44  u32 append(const u32 l, const u8 nl, const u32 r, const u8 nr) {
```

```

45         return (r << nr) | 1;
46     }
47
48     /* extract n bits starting from bit l from x */
49     u32 getbits(const u32 x, const u8 l, const u8 n) {
50         u32 mask = ((1 << n) - 1) << l; /* [l:(l + n)] 1 bits */
51         return (x & mask) >> l;
52     }
53
54     u32 permute_bits(const u32 in, const u8 *permutation, const u8 oplen) {
55         int i;
56         u32 op = 0;
57         u32 temp;
58         for (i = 0; i < oplen; i++) {
59             temp = (in & (1 << permutation[i])) ? (1 << i) : 0;
60             op |= temp;
61             /* permutation[i]th bit of in placed in ith position of out */
62         }
63         return op;
64     }
65
66     void printbits(u32 x, int n) {
67         u32 one_bit = 1;
68         char this_bit;
69         for (int i = 0; i < n; i++) {
70             this_bit = (one_bit & x) ? '1' : '0';
71             putchar(this_bit);
72             one_bit <<= 1;
73         }
74     }
75
76
77     u8 get_round_subkey(u32 key, u8 round_number) {
78
79         /* round 0 entry is dummy */
80
81         /* this is the prefix sum of the rotation array, so that the
82          * "effective" rotation can be used directly */
83         static const u8 shift_routine[] = {0, 1, 3};
84
85         u32 p10_x = permute_bits(key, P10, P10len);
86
87         u32 left_bits, right_bits, rotated_left, rotated_right, combination, final;
88
89         left_bits = getbits(p10_x, 0, 5);
90
91         right_bits = getbits(p10_x, 5, 5);
92
93         rotated_left = rotate_l(left_bits, 5, shift_routine[round_number]);
94
95         rotated_right = rotate_l(right_bits, 5, shift_routine[round_number]);
96
97         combination = append(rotated_left, 5, rotated_right, 5);
98
99         final = permute_bits(combination, P8, P8len);
100
101         return final;
102
103         /* return permute_bits(

```

```

104         append(
105             rotate_l(
106                 getbits(
107                     p10_x,
108                     0,
109                     5),
110                 5,
111                 shift_routine[round_number]),
112             5,
113             rotate_l(
114                 getbits(
115                     p10_x,
116                     5,
117                     5),
118                 5,
119                 shift_routine[round_number]),
120             5),
121         P8,
122         P8len);
123     */
124 }
125
126 void debugprint(char *string, u32 val, u8 bits) {
127     printf("%s: ", string); printbits(val, bits); printf("\n");
128 }
129
130 /* takes 4 bits as input
131  * expands to 8 bit
132  * divides into 2 4 bit parts
133  * uses S-boxes to convert them to 2 bits each
134  * permutes the 4 bit sequence obtained
135  * returns result
136  */
137 u8 T(u8 x, u32 key, u8 round) {
138     u8 permuted, boxin, constructed, row, col, key_combined, k_i;
139
140     const static u8 S[2][4][4] = {
141         /* S[0] */
142         {
143             {1, 0, 3, 2},
144             {3, 2, 1, 0},
145             {0, 2, 1, 3},
146             {3, 1, 3, 2}
147         },
148         /* S[1] */
149         {
150             {0, 1, 2, 3},
151             {2, 0, 1, 3},
152             {3, 0, 1, 0},
153             {2, 1, 0, 3}
154         }
155     };
156
157     const static u8 P4[] = {1, 3, 2, 0};
158     const static u8 P4len = 4;
159
160     /* permute bits of x to get an 8-bit string */
161     permuted = permute_bits(x, EP, EPlen);
162

```

```

163     k_i = get_round_subkey(key, round);
164
165     key_combined = permuted ^ k_i;
166
167
168     constructed = 0;
169
170     for (int i = 0; i < 2; i++) {
171
172         /* select 4 bits from boxin */
173         boxin = getbits(key_combined, i * 4, 4);
174
175         row = (getbits(boxin, 0, 1) << 1) | getbits(boxin, 3, 1);
176
177         col = (getbits(boxin, 1, 1) << 1) | getbits(boxin, 2, 1);
178
179         /* row is chosen based on bits 0 and 3 */
180
181         /* choose proper S-box, get substitution result, append to
182         * current constructed result */
183         constructed = append(constructed, 2, S[i][row][col], 2);
184     }
185
186     u8 p4_x = permute_bits(constructed, P4, P4len);
187
188
189     return p4_x;
190 }
191
192 u8 pi_operation(u8 in, u32 key, u8 round) {
193
194     u8 x, xdash;
195
196     /* first 4 bits */
197     x = getbits(in, 0, 4);
198
199     /* last 4 bits */
200     xdash = getbits(in, 4, 4);
201
202     /* (X + Ti(X'), X') */
203     return append(
204         x ^ T(xdash, key, round),
205         4,
206         xdash,
207         4);
208 }
209
210 u8 sdes_encrypt(u8 in, u32 key) {
211
212     u8 permuted, first_pi, shifted, second_pi, unpermuted;
213
214     permuted = permute_bits(in, IP, IPlen);
215
216     first_pi = pi_operation(permuted, key, 1);
217
218     shifted = rotate_l(first_pi, 8, 4);
219
220     second_pi = pi_operation(shifted, key, 2);
221

```

```

222         unpermuted = permute_bits(second_pi, invIP, invIPlen);
223
224         return unpermuted;
225     }
226
227     u8 sdes_decrypt(u8 in, u32 key) {
228
229         u8 permuted, second_pi, shifted, first_pi, unpermuted;
230
231         permuted = permute_bits(in, IP, IPlen);
232
233         second_pi = pi_operation(permuted, key, 2);
234
235         shifted = rotate_l(second_pi, 8, 4);
236
237         first_pi = pi_operation(shifted, key, 1);
238
239         unpermuted = permute_bits(first_pi, invIP, invIPlen);
240
241         return unpermuted;
242     }
243
244     /* based on:
245      * A SIMPLIFIED DATA ENCRYPTION STANDARD ALGORITHM
246      * Edward. F. Schaefer */
247
248     int main(int argc, char *argv[]) {
249         if (argc != 5) {
250             fprintf(stderr, "usage: sdes <mode> <key> <input_file> <output_file>\n");
251             return EINVAL;
252         }
253
254         u32 key = atoi(argv[1]) % 1024;
255         u8 in, op, de;
256
257         key = atoi(argv[2]) % 256;
258         int fi, fo;
259
260         char mode = argv[1][0];
261
262         /* verify for all possible combinations */
263         /* for (int j = 0; j < 1024; j++) {
264
265             for (int i = 0; i < 256; i++) {
266                 key = j;
267                 in = i;
268                 op = sdes_encrypt(in, key);
269
270                 de = sdes_decrypt(op, key);
271
272                 if (in != de) {
273                     printf("wrong: "); printbits(in, 8); printf("\tseen is: "); printbits(de, 8); printf("\n");
274                 }
275             }
276         } */
277
278         if ((fi = open(argv[3], O_RDONLY)) == -1) {
279             perror(argv[3]);
280

```

```

281         return errno;
282     }
283
284     if ((fo = open(argv[4], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR)) == -1) {
285         perror(argv[4]);
286         return errno;
287     }
288
289     while(read(fi, &in, 1) == 1) {
290         if (mode == 'e') {
291             op = sdes_encrypt(in, key);
292         }
293         else if (mode == 'd') {
294             op = sdes_decrypt(in, key);
295         }
296         else {
297             break;
298         }
299         if (write(fo, &op, 1) != 1) {
300             perror("write");
301             return errno;
302         }
303     }
304
305     close(fi);
306     close(fo);
307
308     return 0;
309 }

```

Output

```

Terminal-
File Edit View Terminal Tabs Help
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes ./sdes
usage: sdes <mode> <key> <input_file> <output_file>
x akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes vi input.txt
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes cat input.txt
Life is paradoxially coincidental to the ironical tyranny applicable to the
unparalleled definition of reverse entropy.
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes ./sdes e 42 input.txt output.txt
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes cat output.txt
W-e0W)05U=U!o7WU##S0yow/yW!e/%U#0%o0%3e0W=0/WyU#0%S=U//S0U55#WyU9#e0%o0%3e:a/5U=U##e#e!0!e-W/W%Wo/0o-0=e)e=e0e/%=o5S
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes ./sdes d 42 output.txt decrypted.txt
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes cat decrypted.txt
Life is paradoxially coincidental to the ironical tyranny applicable to the
unparalleled definition of reverse entropy.
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes diff input.txt decrypted.txt
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes # The two files are the same
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes

```

Figure 1: Example text file encryption and decryption

```

Terminal -
File Edit View Terminal Tabs Help

Untitled x Untitled x

akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes #SDS USED FOR ENCRYPTING THE SDS PAPER PDF FILE
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes time ./sdes e 42 Simplified_DES.pdf sdes_encrypted.bin

real    0m2.506s
user    0m0.779s
sys     0m1.724s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes #SDS USED FOR DECRYPTING THE ENCODED SDS PAPER
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes time ./sdes d 42 sdes_encrypted.bin sdes_decrypted.pdf

real    0m2.559s
user    0m0.769s
sys     0m1.783s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes diff Simplified_DES.pdf sdes_decrypted.pdf
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes echo $?
0
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes ls -l Simplified_DES.pdf sdes_decrypted.pdf sdes_encrypted.bin
-rw-r--r-- 1 akshay akshay 558257 Oct  5 03:08 sdes_decrypted.pdf
-rw-r--r-- 1 akshay akshay 558257 Oct  5 03:08 sdes_encrypted.bin
-rw-r--r-- 1 akshay akshay 558257 Sep 28 14:40 Simplified_DES.pdf
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes

```

Figure 2: Encryption and decryption of pdf file, and time needed using “time” command

```

Terminal -
File Edit View Terminal Tabs Help

akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes time ./sdes e 42 Simplified_DES.pdf enc1.bin

real    0m2.916s
user    0m0.900s
sys     0m1.952s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes time ./sdes e 67 Simplified_DES.pdf enc2.bin

real    0m2.808s
user    0m0.847s
sys     0m1.888s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes time ./sdes e 99 Simplified_DES.pdf enc3.bin

real    0m2.764s
user    0m0.937s
sys     0m1.738s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes time ./sdes e 18 Simplified_DES.pdf enc4.bin

real    0m2.767s
user    0m0.849s
sys     0m1.878s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes python3
Python 3.8.5 (default, Sep  5 2020, 10:50:12)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> (0.9 + 0.847 + 0.937 + 0.849) / 4
0.8832500000000001
>>>

```

Figure 3: Encryption times

```
Terminal -
File Edit View Terminal Tabs Help
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ time ./sdes d 42 enc1.bin dec1.pdf
real    0m2.842s
user    0m0.798s
sys     0m2.005s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ time ./sdes d 67 enc2.bin dec2.pdf
real    0m3.161s
user    0m1.029s
sys     0m2.062s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ time ./sdes d 99 enc3.bin dec3.pdf
real    0m2.685s
user    0m0.849s
sys     0m1.807s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ time ./sdes d 18 en4.bin dec4.pdf
en4.bin: No such file or directory
real    0m0.002s
user    0m0.000s
sys     0m0.002s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ time ./sdes d 18 enc4.bin dec4.pdf
real    0m2.753s
user    0m0.875s
sys     0m1.844s
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ diff Simplified_DES.pdf dec1.pdf
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ diff Simplified_DES.pdf dec2.pdf
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ diff Simplified_DES.pdf dec3.pdf
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ diff Simplified_DES.pdf dec4.pdf
akshay@akshay-inspiron5423 ~/.../lab_assignments/4_sdes$ python3
Python 3.8.5 (default, Sep  5 2020, 10:50:12)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> (0.798 * 1.029 + 0.849 + 0.875) / 4
0.6362855
>>> (0.798 + 1.029 + 0.849 + 0.875) / 4
0.88775
>>>
```

Figure 4: Decryption times

Statistics

The file used for encryption is a pdf file having size **546KB**

The average time needed for encryption (4 repetitions) is **0.8832s**.

The average time needed for decryption (4 repetitions) is **0.8875s**.