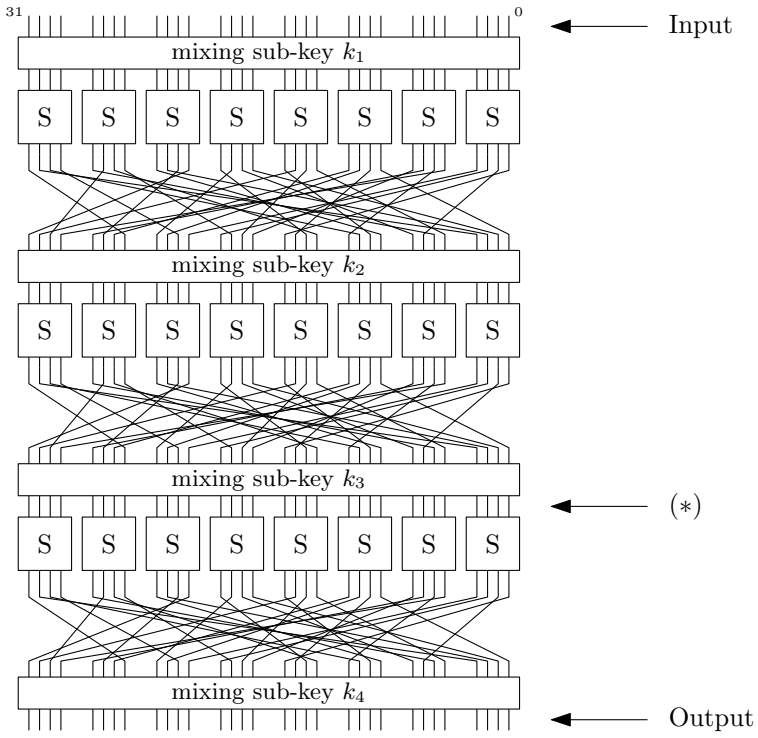
Prof. Martin Hofmann
Dr. Ulrich Schöpp

Ludwig-Maximilians-Universität München
Institut für Informatik
Summer 2017

Cryptography

Exercise Sheet 6

In this Tutorial, we work through differential cryptanalysis for an S/P-network with four rounds. We consider a network as shown below. The last round (xor-ing with sub-key k_4 and after) omits S-Boxes and permutation, as these steps could be inverted anyway.



We write inputs and outputs as hexadecimal numbers, where bit 0 is the least significant bit. For example, the number `0x12345678` means the following list of bits `0001 0010 0011 0100 0101 0110 0111 1000`, from left to right in the figure.

The S-box S is defined by:

input	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
output	e	4	0	1	2	f	8	b	3	a	6	c	5	9	d	7

The permutation in all rounds is given by: `[15 6 19 20 28 11 27 16 0 14 22 25 4 17 30 9 1 7 23 13 31 26 2 8 18 12 29 5 21 10 3 24]`. This means that the permutation

maps bit `0` to bit `15`, bit `1` to bit `6`, and generally bit `i` to the bit whose number appears at the `i`-th position in this list.

While we do not know the key (k_1, k_2, k_3, k_4), we are allowed to encrypt arbitrary messages using the network.

Exercise 6-1 Suppose the network maps two input words to output words `0xffff7fff` and `0xffe77fbf` respectively. We do not know the intermediate values at point $(*)$ for these two inputs. What can you say about the differential of these intermediate values? List all possibilities.

Solution (Sketch): Write $f_p \colon \{0,1\}^{32} \to \{0,1\}^{32}$ for the functions that perform the permutation. Write $f_S \colon \{0,1\}^{32} \to \{0,1\}^{32}$ for the function that applies eight S-Boxes in parallel.

We need to compute the differential of $f_S^{-1}(f_p^{-1}(\texttt{0xffff7fff} \oplus k_4))$ and $f_S^{-1}(f_p^{-1}(\texttt{0xffe77fbf} \oplus k_4))$. Note that we have.

$$\begin{aligned} f_S^{-1}(f_p^{-1}(\texttt{0xffff7fff} \oplus k_4)) &= f_S^{-1}(f_p^{-1}(\texttt{0xffff7fff}) \oplus f_p^{-1}(k_4)) \\ &= f_S^{-1}(\texttt{0xfffffffe} \oplus f_p^{-1}(k_4)) \end{aligned}$$

$$\begin{aligned} f_S^{-1}(f_p^{-1}(\texttt{0xffe77fbf} \oplus k_4)) &= f_S^{-1}(f_p^{-1}(\texttt{0xffe77fbf}) \oplus f_p^{-1}(k_4)) \\ &= f_S^{-1}(\texttt{0xfffffff0} \oplus f_p^{-1}(k_4)) \end{aligned}$$

The differential of the two arguments for f_S^{-1} is:

$$\texttt{0xfffffffe} \oplus f_p^{-1}(k_4) \oplus \texttt{0xfffffff0} \oplus f_p^{-1}(k_4) = \texttt{0x0000000e}$$

Note that it does not depend on the key.

This means that the first seven digits of both words are the same. Since S is a permutation, also their image under S^{-1} must be the same. Note also that f_S^{-1} works by applying S^{-1} eight times in parallel (i.e. for each hexadecimal digit).

In the last digit the two words are different. Since S is a permutation, it must map these two digits to different values.

After xor-ing with $f_p^{-1}(k_4)$, the last digit of the word can be any pair with differential `e`. The S-box is defined such that exactly the inputs with differential `2`, `6`, `8`, `b`, `c` and `f` have output differential `e`.

The possible differentials at $(*)$ are therefore `0x00000002`, `0x00000006`, `0x00000008`, `0x0000000b`, `0x0000000c` and `0x0000000f`, depending on the key k_4.

Exercise 6-2 The following has been verified heuristically: If the differential of inputs to the network is `0x0000b000`, then the most likely differential at point $(*)$ is `0x00800002`. This differential is observed there with probability 0.125. All other differentials at this point have lower probability.

With this knowledge, differential cryptanalysis allows us to (likely) recover eight bits of k_4. We will try out all choices for k_4 that in binary have the form

`*000 0*00 000* *000 *000 000* 0*00 0*00`,

where `*` denotes an arbitrary bit value. These are the bits that connect to the two S-Boxes at $(*)$ that get a non-zero differential input by the differential `0x00800002`. (This means that one gets the bit pattern for k_4 by applying the permutation to the bit pattern `0000 0000 **** 0000 0000 0000 0000 ****`.)

For each of the 256 key candidates for k_4 we keep a counter, initially set to 0, that indicates how likely the candidate is to be correct.

We then generate 100 random input pairs with differential `0x0000b000`. For each such pair (x_1, x_2), we do the following:

- Encrypt x_1 and x_2, to obtain output values y_1 and y_2.
- For each of our 256 candidates of k_4, we invert the network up to point $(*)$ and compute the value there, first starting from the output value y_1, and then starting from the output value y_2. If these two values have a differential of `0x00800002`, then we increment the counter for the current candidate of k_4.

We expect that at point $(*)$ the difference `0x00800002` appears with probability 0.125, i.e. for about 12 of our 100 samples. For the correct key candidate for k_4, we get the true differential at point $(*)$, so we expect the counter for this key to be about 12. For wrong candidates we expect lower counters (at worst the same). So the most probable choice of the key candidate is the one with the highest counter.

Try this out to see if you can recover the key bits.

Solution (Sketch): See program `exercise6_2.c`.

Exercise 6-3 Compute the table of differentials for the given S-Box S and consider how one may compute the most likely difference at point $(*)$ for any given input differential for the network.

Solution (Sketch): See program `exercise6_3.c` for a program that computes the differential table.

```
     INPUT
   |  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
 --+ ------------------------------------------------
I 0| 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
U 1|  0  2  0  2  0  0  0  0  0  2  6  0  2  2  0  0
T 2|  0  0  0  0  2  4  2  0  2  0  2  0  0  0  4  0
P 3|  0  0  2  0  4  0  0  2  0  2  0  0  2  0  0  4
```

```
U 4|  0  0  0  2  0  0  2  0  2  0  2  6  2  0  0  0
T 5|  0  6  0  0  0  0  2  0  0  2  2  2  0  0  0  2
  6|  0  0  2  2  0  2  2  0  0  0  0  0  0  2  4  2
  7|  0  0  0  2  2  2  0  2  0  2  0  0  2  0  0  4
  8|  0  0  0  0  0  2  4  2  0  0  0  0  2  4  2  0
  9|  0  0  0  0  2  0  2  4  0  0  2  2  2  0  0  2
  a|  0  0  2  2  0  0  0  0  6  0  0  2  0  2  0  2
  b|  0  2  8  0  0  2  0  0  0  0  2  0  0  0  2  0
  c|  0  2  0  0  0  2  2  2  0  0  0  2  0  4  2  0
  d|  0  2  0  0  2  0  0  4  2  0  0  0  4  2  0  0
  e|  0  2  0  6  2  2  0  0  2  0  0  2  0  0  0  0
  f|  0  0  2  0  2  0  0  0  2  8  0  0  0  0  2  0
```

By looking at the table, we know that input `b` is most likely mapped to output `2` (with probability 8/16).

So with input differential `0x0000b000`, we expect to see differential `0x00002000` after the first layer of S-Boxes. The permutation maps this value to `0x00020000`.

The table tells us that input `2` is most likely mapped to `5` or `e`, to both with probability 4/16. In the first case, the differentials after the second round of S-Boxes will be `0x00050000`. This is mapped by the permutation to `0x00800002`.

If we assume that both steps are independent (which seems to work in practice), then we get that with input differential `0x0000b000`, the differential at point $(*)$ will be `0x00800002` with probability $(8/16) \cdot (4/16) = 1/8 = 0.125$.

One can simply try out all possibilities to check that there is no other differential with higher probability (but there is a second differential with the same probability). Note that one also has to look at the cases where the first step does not go from `0x0000b000` to `0x00002000`, but to some other value, e.g. `0x00001000`. While this step has a smaller probability (according to the table), it could be that the next following steps have very high probability, leading to a higher probability overall. In this example case, this does not happen, however.

One can write a program to compute the most likely differential by trying out all such all possibilities. There are many ways of cutting down the search space. A simple implementation is used in `differential.c`.