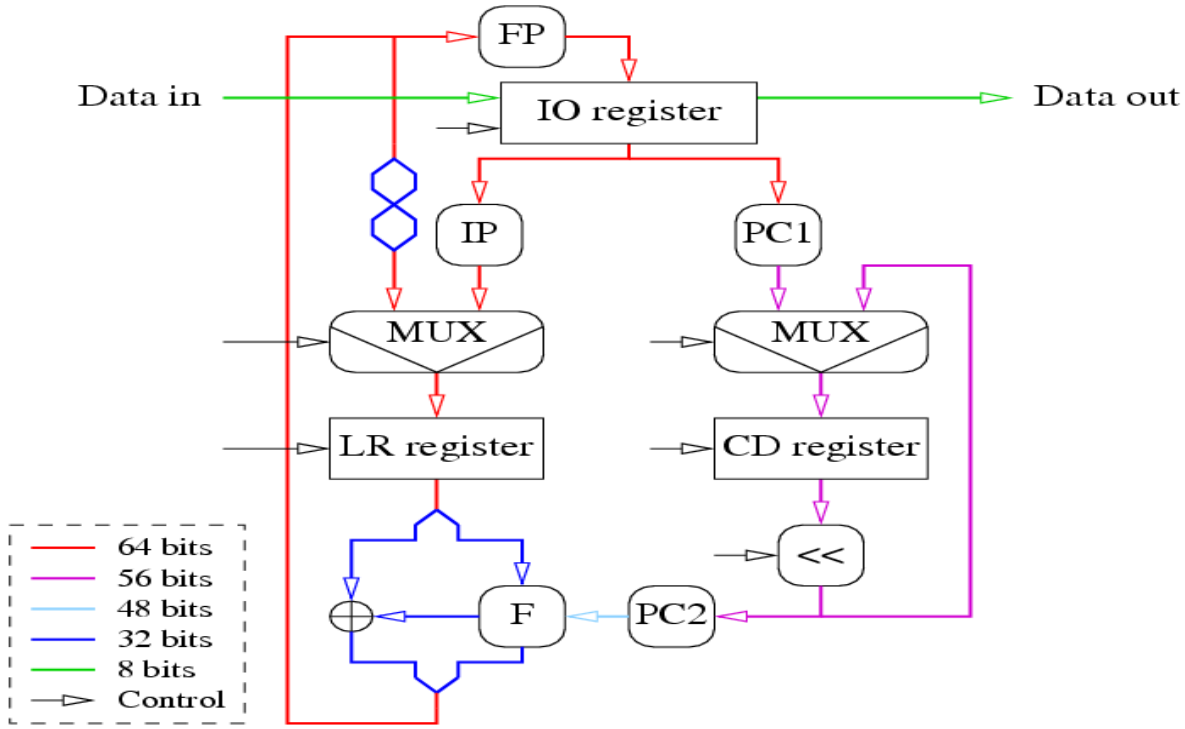
Differential Power Analysis of a DES cryptoprocessor

Lab report

Zhang Yiyi

- **Lab Description:**

In this lab we will try to retrieve a DES last round key from a set of power traces. The target implementation is the hardware DES crypto-processor which architecture is depicted in the following figure:



Using this architecture, **10000** different 64 bits messages were enciphered with **the same secret key**. The power traces were recorded along with the plain texts and the produced cipher texts. The secret key is known. Each DES processing has been recorded 64 times and the 64 power traces were then **averaged** in order to increase the resolution.

- **Lab Solution:**

The original attack model given by course is **Hamming Weight Model**. The idea in the Hamming Weight Model is that in the last round encryption, if the bit_i in L15 of position i is 1, then the program consumes more power than when bit_i is 0. The attack model separates the power traces into two sets according to it. The set_1 consumes more power than set_0 **if the sub key is right**. So the attack model chooses the sub key with the **highest value** of (set_1 – set_0).

However, the weakness of this model is, using only one target bit to choose the corresponding 6 bits sub key is not always accurate. For the same group of 6 bits we should have 4 best guesses, but they are not identical, which means this method is influenced by signal noises. As a result, 10000 power traces is not enough to retrieve the right key. In order to make it more accurate and reduce the number of power traces to use, I try to use **4 bits** from the **same sandbox** to choose the corresponding 6 bits sub key. Instead of looking for the highest peak of one bit, I try to find the highest peak among the 4 bits. The result is slightly better. I can find 3 good sub keys among the 8 sub keys with 10000. But this is far more from a good success.

To move on, I use **Hamming Distance Model** to increase the accuracy. Hamming Distance Model depends on the fact that the **transition** of bit will consume power. No matter from 0 ->1 or 1 ->0, it consumes the same power. Thus, I focus on the LR register, because for now, we can compute L15 with our guess of key, R14 = L15, R15 = L16. So the **flip of bit** is R14 XOR R15. If flip is true, make the power trace in Set_1, else make it in Set_0, as in Hamming Weight model, find the sub key with the highest difference between Set_1 and Set_0. With Hamming Distance Model, it's very efficient to find the good last round key with 1200 power traces.

- **Lab Result:**

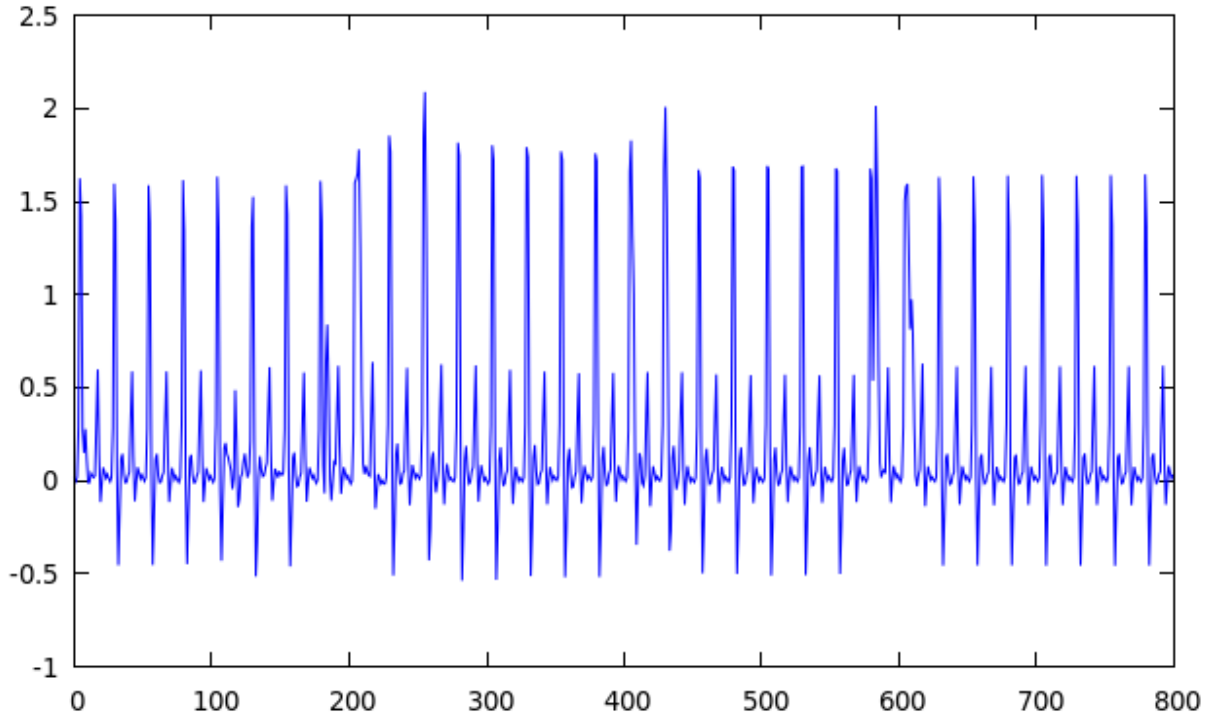
$ cat pa.key

```
Secret key:
  64-bits key (with parity bits):    0x6a65786a65786a65
  56-bits key (without parity bits):   0x00ffff249926d4
  48-bits round key  1 - 6-bits subkeys: 0xe0bee600d677 - 0x38 0x0b 0x3b 0x26 0x00 0x0d
0x19 0x37
  48-bits round key  2 - 6-bits subkeys: 0xe0b676e85988 - 0x38 0x0b 0x19 0x36 0x3a 0x05
0x26 0x08
  48-bits round key  3 - 6-bits subkeys: 0xe4de7600723f - 0x39 0x0d 0x39 0x36 0x00 0x07
0x08 0x3f
  48-bits round key  4 - 6-bits subkeys: 0xe6f372f718a0 - 0x39 0x2f 0x0d 0x32 0x3d 0x31
0x22 0x20
  48-bits round key  5 - 6-bits subkeys: 0xaed773800b7b - 0x2b 0x2d 0x1d 0x33 0x20 0x00
0x2d 0x3b
  48-bits round key  6 - 6-bits subkeys: 0xef535b17ba14 - 0x3b 0x35 0x0d 0x1b 0x05 0x3b
0x28 0x14
  48-bits round key  7 - 6-bits subkeys: 0x2fd3d97105f0 - 0x0b 0x3d 0x0f 0x19 0x1c 0x10
0x17 0x30
  48-bits round key  8 - 6-bits subkeys: 0x1f59db09a80f - 0x07 0x35 0x27 0x1b 0x02 0x1a
0x20 0x0f
  48-bits round key  9 - 6-bits subkeys: 0x1f4bd93007e9 - 0x07 0x34 0x2f 0x19 0x0c 0x00
0x1f 0x29
  48-bits round key 10 - 6-bits subkeys: 0x1f799d9ab803 - 0x07 0x37 0x26 0x1d 0x26 0x2b
0x20 0x03
  48-bits round key 11 - 6-bits subkeys: 0x1f2dcd666730 - 0x07 0x32 0x37 0x0d 0x19 0x26
0x1c 0x30
  48-bits round key 12 - 6-bits subkeys: 0x5b6cad39294a - 0x16 0x36 0x32 0x2d 0x0e 0x12
0x25 0x0a
  48-bits round key 13 - 6-bits subkeys: 0xd9adace4d012 - 0x36 0x1a 0x36 0x2c 0x39 0x0d
0x00 0x12
  48-bits round key 14 - 6-bits subkeys: 0xd0aeaf45266e - 0x34 0x0a 0x3a 0x2f 0x11 0x12
0x19 0x2e
  48-bits round key 15 - 6-bits subkeys: 0xf1be26bc98c8 - 0x3c 0x1b 0x38 0x26 0x2f 0x09
0x23 0x08
  48-bits round key 16 - 6-bits subkeys: 0xf0be2e5b242c - 0x3c 0x0b 0x38 0x2e 0x16 0x32
0x10 0x2c
```

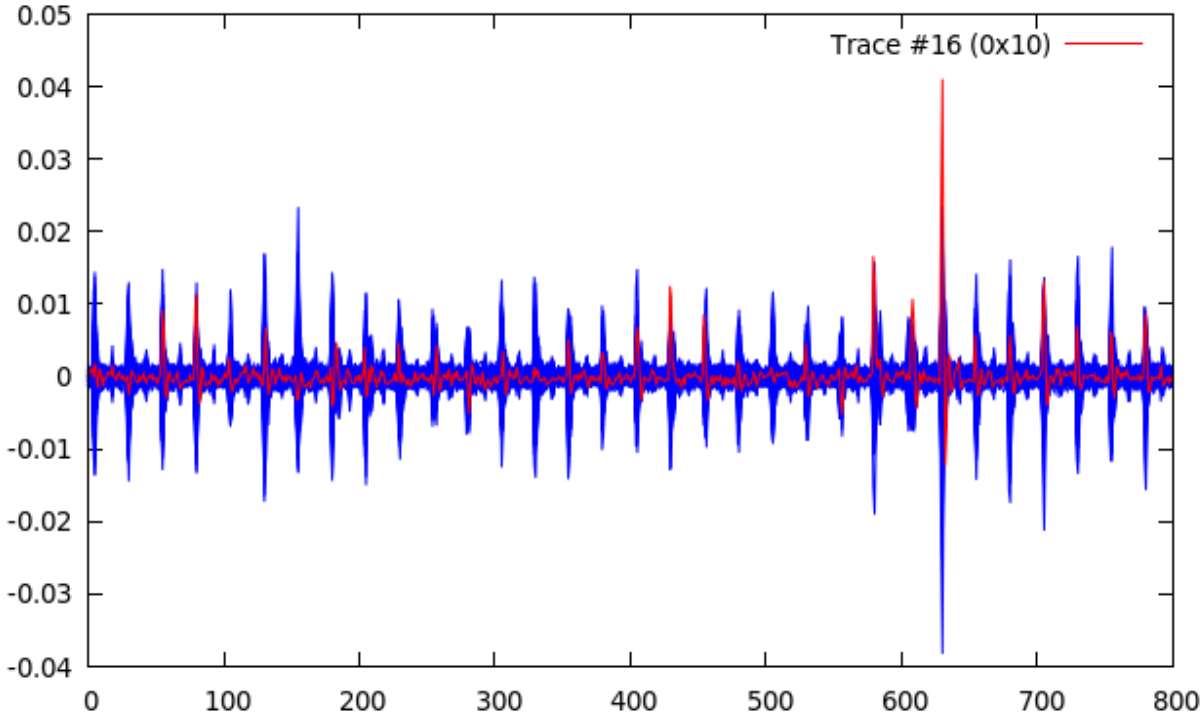
$ ./pa pa.dat 1200

```
Last round key (hex):
0xf0be2e5b242c
```

$ gnuplot -persist average.cmd
$ gnuplot-pesisist dpa.cmd



The figure above represents such a power trace with the time as horizontal axis and the power (instantaneous current) as vertical axis. We can find that there are **32** clock periods in total, with 8 cycles to input the message, 16 cycles to compute the 16 rounds of DES, 8 cycles to output the result. So **Simple Power Analysis** is already powerful to reveal the sequence of process very clearly, because different processes consume different power. Since the trace has been averaged, the influence of noises will be reduced.



The picture above is the **differential traces**. 63 wrong sub keys are drawn in blue, the **1 right** sub key is drawn in **red**. The sandbox number is **7**, so the 6 bits sub key is **0x10**, so it is the right sub key. We can easily detect that the red plot has the highest peak. The peak of the red line is about 0.04, which is the highest difference between the average power trace. If we think more, we will discover that the index is **630**, which is exactly the time when the flip of the bits happens. So it verifies that **Hamming Distance Model** can really work.

Lab Conclusion:

We can find that Differential Power Analysis is very powerful. But there is another way to improve the accuracy of the guess, which is **Pearson Correlation Coefficient**. The idea is that the hamming weight is **proportional** to the power consumption. As for the 4 target bits from the same sandbox, add their power consumption together and find the highest peak. If the sub key is correct, we will get a high pcc number between hamming weight of the 4 target bits and the highest power consumption. So choose the sub key with the **highest** pcc number will be a good solution.

However, is there any **countermeasure** to DPA? Referring to the paper from Paul Kocher about DPA, there are three ways to prevent DPA. One is to reduce signal sizes, choose operations that leak less information in power consumption. The second is to insert some noises into power consumption measurements. The final one is to design cryptosystem to ensure that power traces cannot be related to transactions. However, the first and second are not safe enough if there are large number of samples.