Side Channel Attacks (SPA)

on HaHa v3.0 Board

We will implement a Simple Power Analysis (SPA) side channel attack on a modular exponentiation circuit.

Instructor: Dr. Swarup Bhunia
Co-Instructors/TAs: Shuo Yang and Reiner Dizon-Paradis

Theory Background

1. Side channel attack techniques

In cryptography, a side channel attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms (compare cryptanalysis). For example, timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited to break the system. Some side-channel attacks require technical knowledge of the internal operation of the system on which the cryptography is implemented, although others such as differential power analysis are effective as black-box attacks. Many powerful side-channel attacks are based on statistical methods pioneered by Paul Kocher.

1.1 Ways of accessing the module

When analyzing the security of a cryptographic hardware module, it can be useful to perform a systematic review of the attack surface — the set of physical, electrical and logical interfaces that are exposed to a potential opponent. According to this observation, side channel attacks can be divided into the following classes: invasive attacks, semi-invasive attacks, and non-invasive attacks.

An Invasive attack involves de-packaging to get direct access to the internal components of cryptographic modules or devices. A typical example of this is that the attackers may open a hole in the passivation layer of a cryptographic module and place a probing needle on a data bus to see the data transfer. The concept of semi-invasive attack is first developed by Skorobogatov and Anderson. This kind of attack involves access to the device, but without damaging the passivation layer or making electrical contact other than with the authorized surface. For example, in a fault-induced attack, the attacker may use a laser beam to ionize a device to change some of its memories and thus change the output of this device. A non-invasive attack involves close observation or manipulation of the device's operation. This attack only exploits externally available information that is often unintentionally leaked. A typical example of such an attack is timing analysis: measuring the time consumed by a device to execute an operation and correlating this with the computation performed by the device in order to deduce the value of the secret keys.

1.2 Known side channel attacks

1.2.1 Power attacks

In this experiment, we will focus on power attacks. Power attacks measure the circuit's processing time and current consumption to infer what is going on inside it. Of all types of SCA attacks known today, the number of literature on power analysis attacks and the relevant countermeasures is the biggest. Power analysis attack is the current research focus of side-channel attacks. Power analysis attacks have been demonstrated to be very powerful attacks for most straightforward implementations of symmetric and public key ciphers.

Basically, power analysis attack can be divided into SPA and DPA. In SPA attacks, the aim is essential to guess from the power trace which particular instruction is being executed at a certain time and what values the input and output have. Therefore, the adversary needs an exact knowledge of the implementation to mount such an attack. On the other hand, DPA attack does not need the knowledge of the implementation details and alternatively exploiting statistical methods in the analysis process. DPA is one of the most powerful SCA attacks, yet it can be mounted using very little resources.

1.2.2 Timing attacks

A timing attack is, essentially, a way of obtaining some user's private information by carefully measuring the time it takes the user to carry out cryptographic operations. The principle of this attack is very simple: to exploit the timing variance in the operation.

Timing attacks were introduced in 1996 by Kocher, where RSA modular exponentiation was being attacked. The basic assumptions of timing analysis are 1. The runtime of a cryptographic operation depends to some extent on the key. With present hardware, this is likely to be the case, but note that there are various efficient hardware-based proposals to make the timing attack less feasible through 'noise injection'. Software approaches to make the timing attack infeasible are based on the idea that the computations in two branches of a conditional should take the same amount of time ('branch equalization'). 2.A sufficiently large number of encryptions can be carried out, during which time the key does not change. A challenge-response protocol is ideal for timing attacks. 3.Time can be measured with known error. The smaller the error, the fewer time measurements are required.

1.2.3 EM attacks

As electrical devices, the components of a computer often generate electromagnetic radiation as part of their operation. An adversary that can observe these emanations and can understand their causal relationship to the underlying computation and data may be able to infer a surprising amount of information about this computation and data. This ability can be devastating, should the computer be a trusted computing platform intended to keep this information from the adversary. Similar to the power analysis attacks, Electromagnetic Analysis attacks can also be divided into two main categories: Simple Electromagnetic Analysis and Differential Electromagnetic Analysis.

2. SPA

SPA is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations. SPA can yield information about a device's operation as well as key material.

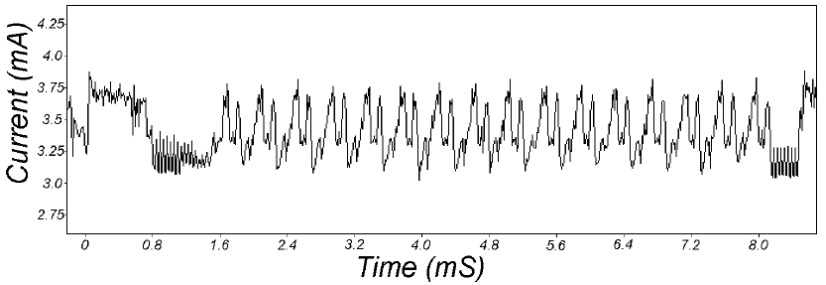

Figure 1 SPA trace showing an entire DES operation.

A trace refers to a set of power consumption measurements taken across a cryptographic operation. For example, a 1-millisecond operation sampled at 5 MHz yields a trace containing 5000 points. Figure 1 shows a SPA trace from a typical smart card as it performs a DES operation. Note that the 16 DES rounds are clearly visible.

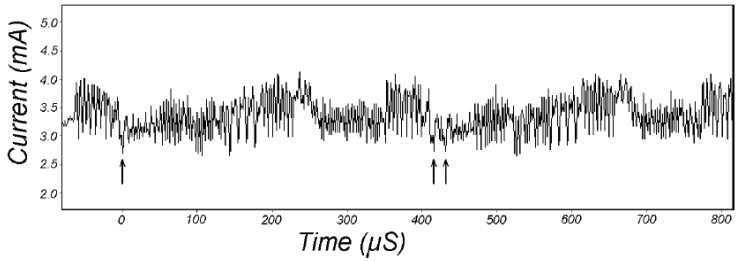

Figure 2 SPA trace showing DES rounds 2 and 3.

Figure 2 is a more detailed view of the same trace showing the second and third rounds of a DES encryption operation. Many details of the DES operation are now visible. For example, the 28-bit DES key registers C and D are rotated once in round 2 (left arrow) and twice in round 3 (right arrows). In Figure 2, small variations between the rounds just can be perceived. Many of these discernable features are SPA weaknesses caused by conditional jumps based on key bits and computational intermediates.

Experiment Set-up: Configuration

The hardware and software needed for this experiment include:

1. The HaHa v3.0 Board.
2. An Oscilloscope
3. GOWIN FPGA Designer to program the FPGA.

Instructions and Questions

```
SquareMult(x, e, N):
  let e_n, ..., e_1 be the bits of e
  y ← 1
  for i = n down to 1 {
    y ← Square(y)                (S)
    y ← ModReduce(y, N)          (R)
    if e_i = 1 then {
      y ← Mult(y, x)             (M)
      y ← ModReduce(y, N)        (R)
    }
  }
  return y
```

You will implement a SPA on square and multiply algorithm, which is often used in DES to implement modular exponentiation functions. The algorithm to calculate "$x^e \bmod N$" is shown in Figure 4. We can see that there is an if-statement controlled by the value of e. If e_i (i^{th} bit of e) is zero, only two steps will be needed: Square and ModReduce. Otherwise, if e_i is one, four steps will be needed: Square, ModReduce, Mult and another ModReduce. Therefore, in the case when e_i is one, the algorithm tends to consume more power (or time, in some cases). We can implement the algorithm into the FPGA and measure how much power it consumes.

Figure 4 Square and multiply algorithm

The code for square and multiply algorithm is provided to you. *mul.v* is the top module. *other.v* contains all the submodules. Create a top module that connects **mul** to the board I/O. Make sure to connect the output to an I/O (eg. LEDs), otherwise the circuit will be optimized away.

A synthesis constraints file is included (synth_constraints.gsc), which will ensure the tool uses the FPGA fabric instead of DSPs to implement the circuit. Make sure and include this in your project. The FPGA does not have enough DSPs to implement this circuit, so you will get an error if you do no include it.

After synthesizing, check the Hierarchy tab to make sure it wasn't optimized away. The **mul** module should require about 1500 LUTs and 2900 ALUs.

Figure 5 shows the sub-circuitry for the HaHa v3.0 board. You can find R17, which is a very precise small resistor on the board. The current consumed by the core of the FPGA will all go through this resistor. Therefore, the more power the FPGA consumes, the voltage drop across the resistor will be bigger.

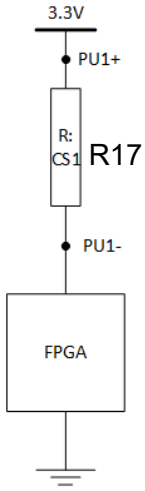

Figure 5 Circuitry

Measuring the voltage difference as shown in Figure 7 and NEVER DO what is shown in Figure 6. The next page shows an example of the voltage when e=8'b11110000.

Figure 6 Wrong way of measuring

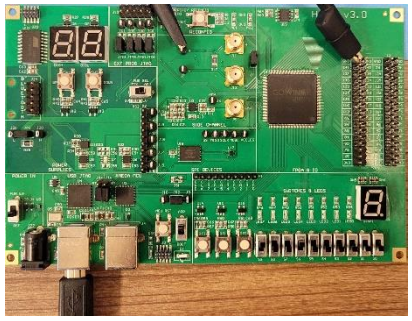

Figure 7 Right way of measuring

Actually, the picture on the right is incorrect as well. It shows probing J8; however, this is the point between the power regulator and the resistor. You actually want to probe J7, which is "after" the resistor (ie. between the resistor and FPGA VCC input).

You will also want to remove jumper JP1, which will bypass the resistor. Make sure to put this jump back after completing the lab.

Change the key (value of e) to see how the power consumption changes.

Write your report to include the following:

1. The input clock (50 MHz) is slowed down by the **divi** module.
    a. What is the period of the new clock (**mul/clk**)?
    b. How many cycles does it take to rotate through the **e** value? Given this, what should be the period of the repeating power usage pattern? *Hint: look at the picture on the next page to verify your answer.*
2. Why should we never do what is shown in Figure 6?
3. Take a picture of the oscilloscope when is 8'b01110001. Zoom appropriately to show a full period of the computation.
4. Try the bitstream files KeyA.fs, KeyB.fs, and KeyC.fs. Guess what value *e* has for each of them. (Don't worry if you guess a "rotated" value of *e*)

Waveform when e is 8'b00001111

References and Further Reading

[1] https://en.wikipedia.org/wiki/Exponentiation_by_squaring

[2] https://en.wikipedia.org/wiki/Data_Encryption_Standard

[3] Kocher, Paul, Joshua Jaffe, and Benjamin Jun. "Differential power analysis." Annual International Cryptology Conference. Springer Berlin Heidelberg, 1999.

[4] Zhou, YongBin, and DengGuo Feng. "Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing." IACR Cryptology ePrint Archive 2005 (2005): 388.

[5] Prouff, Emmanuel. "DPA attacks and S-boxes." International Workshop on Fast Software Encryption. Springer Berlin Heidelberg, 2005.

[6] Guilley, Sylvain, et al. "Silicon-level solutions to counteract passive and active attacks." Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC'08. 5th Workshop on. IEEE, 2008.

[7] Guilley, Sylvain, et al. "Improving side-channel attacks by exploiting substitution boxes properties." International Conference on Boolean Functions: Cryptography and Applications (BFCA). 2007.

[8] Hnath, William. Differential power analysis side-channel attacks in cryptography. Diss. Worcester Polytechnic Institute, 2010.