Thermocommunication

Julien Brouchier[1], Nora Dabbous[2,3], Tom Kean[4], Carol Marsh[4], and David Naccache[5]

[1] Institut supérieur de l'électronique et du numérique
Maison des Technologies - place Georges Pompidou FR-83000, Toulon, France.
julien.brouchier@isen.fr
[2] Telecom ParisTech - Département informatique et réseaux
46, rue Barrault, FR-75634, Paris Cedex 13, France
[3] Ingenico
192 Avenue Charles de Gaulle, FR-92200, Neuilly, France
nora.dabbous@ingenico.com
[4] Algotronix Ltd.
P.O. Box 23116, Edinburgh, EH8 8YB, United Kingdom
tom@algotronix.com, carol@algotronix.co.uk
[5] École normale supérieure, Département d'informatique
45, rue d'Ulm, F-75230, Paris Cedex 05, France.
david.naccache@ens.fr

Abstract. Looking up – you realize that one of the twelve light bulbs of your living room chandelier has to be replaced. You turn electricity off, move a table to the center of the room, put a chair on top of the table and, new bulb in hand, you climb up on top of the chair. As you reach the chandelier, you notice that... all bulbs look alike and that you have forgotten which bulb needed to be changed.

Restart all over again?

Luckily, an effortless creative solution exists. By just touching the light bulbs you can determine the cold one and replace it! Put differently, information about the device's malfunction leaked-out via its temperature...

1 Introduction

In addition to its usual complexity postulates, cryptography silently assumes that secrets can be physically protected in tamper-proof locations. All cryptographic operations are physical processes where data elements must be represented by physical quantities in physical structures. These physical quantities must be stored, sensed and combined by the elementary devices (gates) of any technology out of which we build tamper-resistant machinery. At any given point in the evolution of a technology, the smallest logic devices must have a definite physical extent, require a certain minimum time to perform their function and dissipate a minimal *switching energy* when transiting from one state to another.

Confining a program during its execution so that it cannot leak information to other programs is both an old concern [2] and a difficult task. Recently, several researchers succeeded to fingerprint distant machines by measuring temperature side-effects on clocks [6].

But can temperature also leak secrets *within* a computer or *within* a chip?

2 Process-To-Process Thermocommunication

We started by implementing a covert channel between two processes (a sender and a receiver) running on the same machine. Producing heat is simple: all the sender has to do is launch massive calculations. To sense temperature inside the machine, we considered the following options:

Fan-based solutions: PCs have an internal fan whose (software readable!) angular speed is strongly correlated to the motherboard's temperature. Alternatively, fan speed variations, causing acoustic noise differences, might also be monitored by the machine's microphone.

Built-in sensors: Most hard-disks contain software-readable sensors.

SMART (Self-Monitoring Analysis & Reporting Technology) is an IBM standard for monitoring the disk's status using built-in sensors. Measured parameters are called *attributes*; and attribute 194 is temperature. Extreme heat peaks can also be monitored by reading the Pentium's overheat hardware flag, `IA32_THERM_STATUS`.

Faults as heat detectors: Programs for purposely provoking errors (*e.g.* CPUBurn-in.exe) are commonly used to adjust CPU speed immediately below the hardware failure threshold. Such programs heat the CPU and continuously monitor for erroneous calculations to determine the machine's over-clocking limits. For convert communication purposes, the absence of faults can be interpreted by the receiver as a zero and the presence of faults as a one.

We successfully leaked meaningful data using fan speed measurements. The experiment was successfully repeated on two different hardware platforms: a PC under Debian Linux 2.6.22 (Intel Core2 Duo, 1.80GHz, 2GB RAM) and a MacBookPro under Mac OS X 10.5.2 (Intel Core2 Duo, 2.16GHz, 3GB RAM).

In our experiments, the sender stabilized a sequence of heat levels using Proportional Integral Derivative (PID) regulation.

A PID controller is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired set-point by calculating and then outputting a corrective action that adjusts the process accordingly.

The algorithm involves three parameters: the Proportional, the Integral and Derivative constants. The Proportional constant K_p determines the reaction to the current error, the Integral constant K_i determines the reaction based on the sum of recent errors, and the Derivative constant K_d determines the reaction based on the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process via a control stimulus.

By tuning the three constants, the controller can provide a control action designed for specific process requirements. The controller's response can be described in terms of sensitivity to an error, the degree to which the controller overshoots the set-point and the tolerable degree of system oscillation.

In our case, the PID process, shown in Figure 1, stabilized the fan speed $e(t)$, at a target set-point E chosen by the sender. PID drives the machine's computational load as a weighted sum of $e(t)$, the derivative of $e(t)$ and the integral of $e(t)$ (over a time-period Δ). Our program transmits one bit per three minutes, a rate at which a typical RSA secret key (one 512-bit prime factor) leaks in a day.

Figure 2 shows the fan angular speeds measured by the receiver. Information was modulated by setting the fan speed to 3500, 4000, 4500 or 5000 revolutions per minute (RPM), interpreted by the receiver as $3500 + \text{data} \times 500$.

Figure 2 shows that the sender modulated the data stream `0201 1121 0321 0321 3321`. The receiver accurately measured the stream (each acquisition was averaged during three minutes), reversed it (`1233 1230 1230 1211 1020`), converted it from quaternary to hexadecimal (`0x6F 0x6C 0x6C 0x65 0x48`) and successfully displayed the corresponding ASCII message `HELLO`.

3 Machine-To-Machine Thermocommunication

As we write these lines, we do not know if information can transit between two different PCs inside a closed (unventilated) rack but experiments show that information leaks when nine PCs slotted in the rack's lower decks are simultaneously turned into heat sources to send information to one receiving PC, located in the rack's upmost deck (1 bit per 20 minutes).

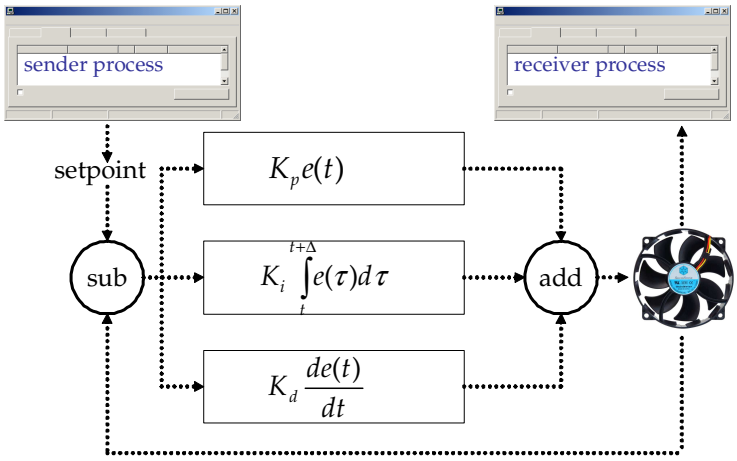

Fig. 1. Proportional Integral Derivative (PID) regulation

4 Circuit-To-Circuit Thermocommunication

In a second experiment, we challenged a hardware protection mechanism called the "Moat & Drawbridge" [1]. Figure 3-A shows a traditional FPGA (Field Programmable Gate Array) containing intellectual property (IP) cores belonging to three different manufacturers (*e.g.* a microprocessor (blue), a cryptographic coprocessor (red) and a network interface (pink)). Circuit synthesis tools "melt" the IP cores into one circuit. Hence each IP core is not guaranteed total control over its internal secrets. The Moat & Drawbridge design, and the mechanism developed by Xilinx and the NSA in [3], consist in using custom synthesis tools that reserve some of the target FPGA's CLBs (Configurable Logic Blocs) as communication channels. The result is shown in figure 3-B. Whenever any of the cores "needs privacy" it "lifts the drawbridge" by disconnecting the communication CLBs depicted in green. This isolation technique was implemented on Virtex-4 FPGAs and found to be satisfactory by [1] and [3].

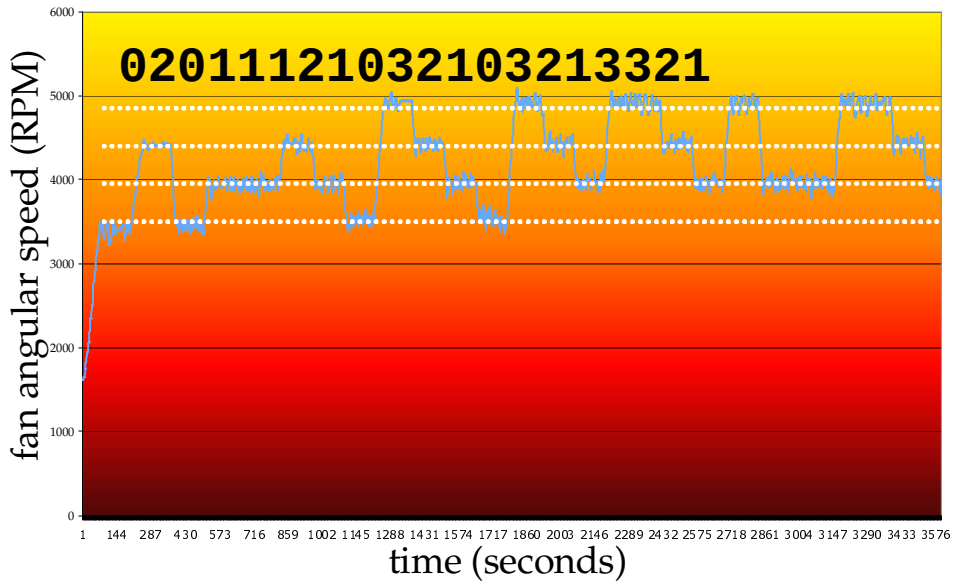

Fig. 2. Temperature-correlated plot (fan speed *vs.* time)

A secret leakage mechanism defeating these protections is depicted in Figure 3-C. The challenge consisted in implementing both a heat source and a heat sensor using purely digital means. Generating heat is easy: we did so using a battery of ring oscillators similar to the one depicted in Figure 4. A ring oscillator is a ring made of an odd number of inverting buffers.

Upon `HeatEnable` signal activation the device starts oscillating, charging and discharging internal conductors and hence generating heat. To sense heat, we also use a ring oscillator.

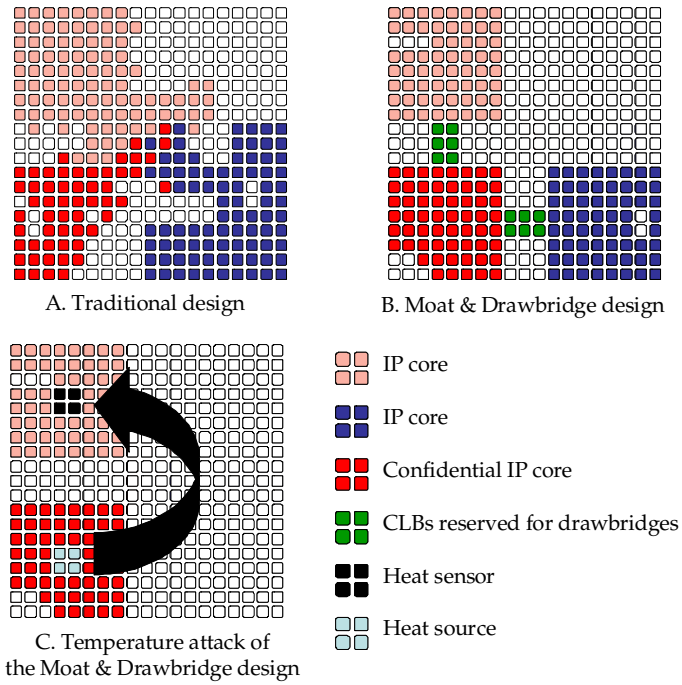

Fig. 3. FPGA synthesis methods)

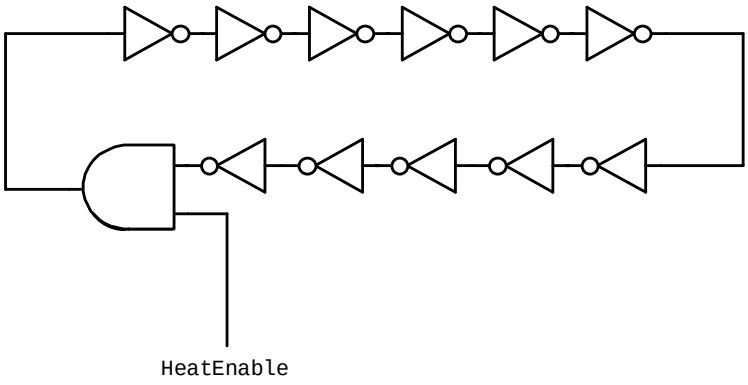

Fig. 4. Ring oscillator

The frequency at which a ring oscillator oscillates is highly temperature-dependent. Hence by using the ring oscillator to clock a long shift-register and by regularly measuring the progress of data in this shift-register, frequency differences (and hence temperature differences) can be sensed. The heat sensor is a small circuit consisting of less than 200 slices in a Xilinx Spartan 3. Experimentally, we achieved a relatively high leakage throughput (1 bit/s), because silicon is very thermally conductive.

5 Thermo-Bugs

We also conducted experiments with the "temperature bug" shown in Figure 5. The user can interact with the bug using two LEDs (1), an infrared interface (2) for data read-back and an activation switch (4). The bug's PIC 16F913 controller (3) stores the temperature acquisitions in a 256K I2C CMOS serial EEPROM (5). The device's mini-battery and quartz (on the printed circuit board's backside) are not shown. The bug can record 16,000 temperature points over 18 hours with a programmable start of recording of up to a year. Measurable temperatures range between $-10°$ and $+70°$. Software running on the machine successfully modulated to the bug, externally attached to the PC's chasing, 80 data bits in 16 hours.

More classic acoustic bugs, listening to fan speed variations could also be used. Note that acoustic threats were considered in the past [5], but in [5] the fan was considered as a nuisance and not as an information source[6].

6 Questions & Further Research

The experiments reported in this paper raise several interesting questions:

- Can overheating faults caused by software be exploited to mount fault attacks [4]?
- From a defensive perspective, can "guard processes" be used to regulate an invariant (or random) internal temperature?
- Can operating systems randomly assign computing bandwidth to threads, thereby preventing any specific thread from controlling alone the motherboard's temperature?
- Is the isothermal coating (or active cooling) of sensitive system parts necessary and realistic?
- Can temperature allow convert communication between applets in Java virtual machine?

References

1. T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen and C. Irvine, *Moats and Drawbridges: An Isolation Primitive for Reconfigurable Hardware Based Systems*, Proceedings of IEEE-SP'07, IEEE Symposium on Security and Privacy, pages 281-295, 2007.

[6] *"...the interesting acoustic signals are mostly above 10KHz, whereas typical computer fan noise is concentrated at lower frequencies and can thus be filtered out by suitable equipment..."*

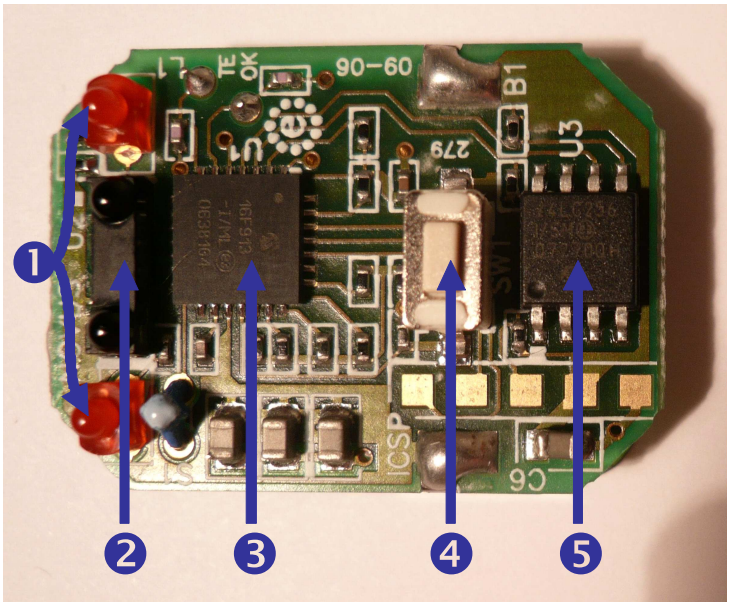

Fig. 5. Miniature temperature bug ($3cm^2$).)

2. B. Lampson, *A Note on the Confinement Problem*, CACM, 16(10), pages 613-615, 1973.
3. M. McLean and J. Moore. FPGA-*based single chip cryptographic solution - securing* FPGAs *for red-black systems*. Military Embedded Systems, March 2007.
4. D. Naccache, *Finding faults*, IEEE Security and Privacy, 3(5), pages 61-65, 2005.
5. A. Shamir and E. Tromer, *Acoustic cryptanalysis - On nosy people and noisy machines*. `http://people.csail.mit.edu/tromer/acoustic`.
6. S. Zander and S. Murdoch, *An Improved Clock-skew Measurement Technique for Revealing Hidden Services*, Proceedings of the 17-th USENIX Security Symposium, pages 211-225, 2008.