Attacking Microcontrollers

Author: Dionisio Perez-Mavrogenis (dpm3g10)

Supervisor: Klaus-Peter Zauner (kpz)

Schoold of Electronics and Computer Science, University of Southampton

Microcontroller Introduction

Microcontrollers can be found anywhere, from your cars stereo to missile launch panels and are usually cheap (around £2) and packed with information! They often come with crypto-engines (AES, DES and RSA are common) and hold all sorts of information like private crypto-keys for authentication or propietary algorithm implementations in the firmware or hardware, interesting all sorts of people into the contents of a microcontroller.

Packaging and De-packaging

Typically microcontrollers are too small and fragile to use as fabricated and so they are packaged. Packacking material ranges depending on the microcontroller and its intended use, but is usually hard epoxy resin [5] [6]. The packaging tries to protect the microcontroller from its external environment (humidity, radiation, temperature, crashes etc.) and also from prying eyes. Military-grade chips come with a lot of additional circuitry on the packaging whose responsibility is to detect tampering and respond in a suitable manner (even self destruction!) [6].

De-packaging is not always requierd and the methods depend on the packaging used and protective mechanisms in place, but on epoxy-packaged chips one can etch the epoxy away by using HNO_3 or H_2SO_4 and then cleaning the chip in an ultrasonic bath [5] [6]. For other packaging types, e.g. metal, ceramic or plastic, one can use similar techniques and tools, e.g. drills or a blowtorch [6].De-packaging is usually easier than expected and removing simple epoxy resin can be done with readily available chemicals [6]. Fig. 1 and Fig. 2 show a microcontroller in its factory resin packaging and the exposed die after chemical decapsulation[7].

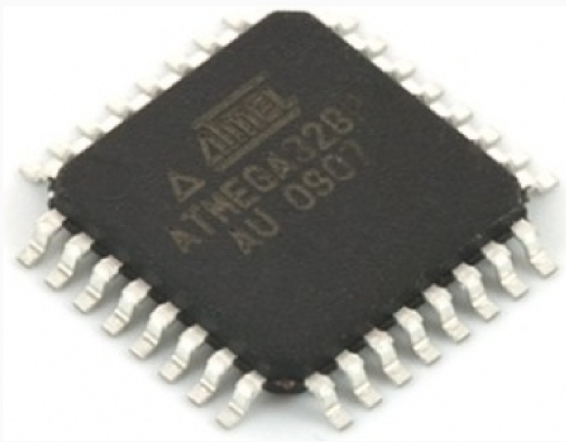

Fig. 1: ATmega328 with epoxy resin packaging.

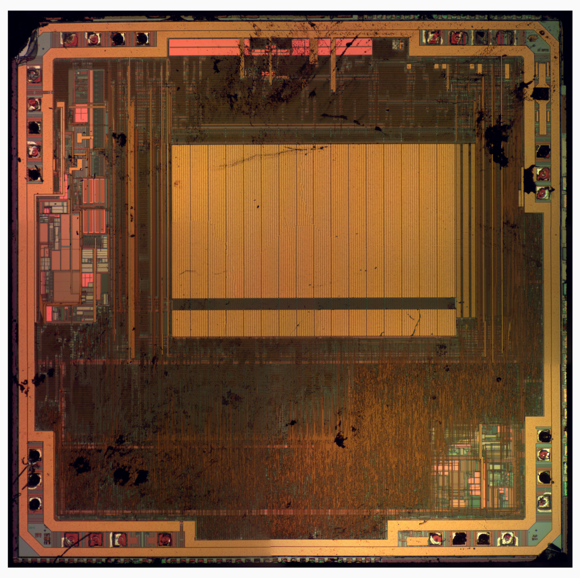

Fig. 2: The exposed die of an ATmega328.

Attack Types

Non-invasive attacks are cheap and easy to perform and require no decapsulation. Popular methods include are power analysis and fault injection, where faults may be injected by exposing the chip to environmental conditions that it was not meant to work in. **Non-Invasive** and **Semi-invasive** attacks are more technical, expensive and lengthier to perform as they require decapsulation and specialized machinery, but yield more information about a die. Attacks under these categories include micro-probing the device, inducing faults using lasers and physical modification of the chip using FIBs [6] [5].

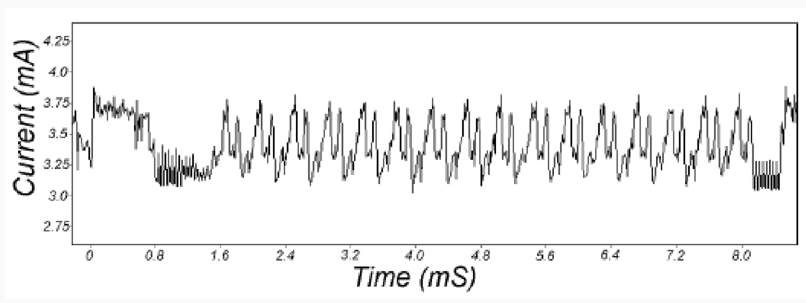

Fig. 3: DPA trace for DES (source:[2]).

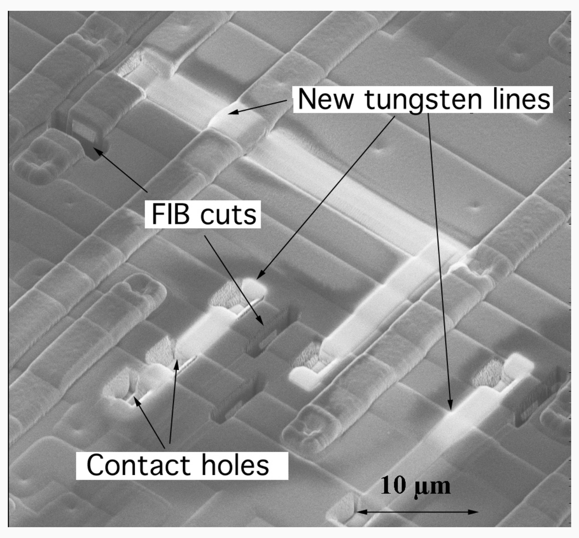

Fig. 4: Chip modification using FIB (source:[3]).

Sample Attack

A simple clock glitch attack could be performed against an ATmega644, in order to make it dump its memory. Since we know how many cycles each instruction takes we would use an FPGA to deliver both the clock signal and the glitch itself, delivering it on the appropriate cycle after a *trigger* event [1][4].

Suppose there's a `while`-loop in the firmware, outputing to a pin via an `OUT` instruction somewhere (Fig. 6). We can count the cycles between successive outputs and use the `OUT` as our trigger, glitching the desired number of cycles after that.

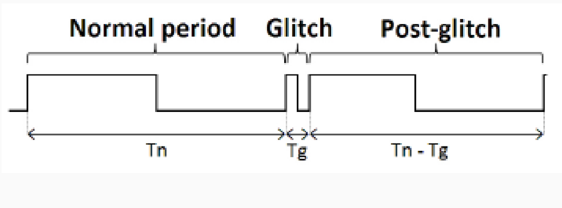

Fig. 5: Sample glitch generation (source: [1]).

```
condition :                              ; label
            SUB R16, R17                 ; n-i, result affects ZF bit
                                           of SREG.
            BREQ leave                   ; if ZF=0 branch to loop,
                                         ; 1 cycle if false, else 2
loop :                                   ; label
            ...code loading from
            memory to variable ...       ; k cycles
            OUT $18, R17                 ; output to port B
            INC R17                      ; increment i, 1 cycle
            JMP condition                ; 3 cycles
leave :                                  ; label
```

Fig. 6: Loop assembly `while(i<n){}`.

References

[1] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus. In *Proceedings of the 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, FDTC '11, pages 105–114, Washington, DC, USA, 2011. IEEE Computer Society.

[2] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[3] S. Leppavuori. Electronics materials, packaging and reliability techniques (empart). `http://www.infotech.oulu.fi/Annual/2000/EMPART.html`, May 2014.

[4] Ramiro Pareja Veredas. Fault injection attacks on microcontrollers: clock glitching tutorial. `http://www.t4f.org/articles/fault-injection-attacks-clock-glitching-tutorial/`.

[5] Sergei P. Skorobogatov. Semi-invasive attacks – A new approach to hardware security analysis. Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, April 2005.

[6] Bulent Yener and Andrew Zonenberg. CSCI 4,74 / 6974 : Hardware Reverse Engineering. Rensselaer Polytechnic Institute, lecture slides at : `http://security.cs.rpi.edu/courses/hwre-spring2014/`, 2014.

[7] Andrew Zonenberg. Atmega328 decapsulation. `http://siliconpr0n.org/archive/doku.php?id=azonenberg:atmel:atmega328`, March 2014.

LaTeX TikZposter