Firmware Protection and Attacks Against the ATmega Microcontrollers

Dionisio Perez-Mavrogenis

Abstract—This paper will review some of the attacks used against microcontrollers in order to obtain access to cryptographic material (keys or algorithms) or the firmware of the device, both of which have serious implications. Defence techniques developed against the attacks will also be reviewed, in an attempt to show the co-evolutionary nature of the attack-defence ecosystem. We conclude by presenting a working attack against a microcontroller and by discussing why completely securing a microcontroller is impossible and that manufacturers should focus on making hacking the microcontroller "hard enough" in terms of money and cost.

Index Terms—Microcontroller attacks, firmware dumping, reverse engineering, microprobing, microcontroller protection, defense engineering

review wherever I say working attack against the atmega to possible attack. reduce structure of things, merge sentences together. also make sure that spaces between Fig. or Table and number are unbreakable using the tilde. also consistently name abbreviations, could have abbreviation table.

1 Introduction

This paper will present an overview of the current attack methods for tampering with MCUs (MicroController Unit) in order to obtain access to IP (Intellectual Property). IP refers to the firmware of the MCU and information that could be obtained from that, i.e. implementation secrets or proprietary algorithms that help a manufacturer achieve higher performance over their competitors. Firmware tampering (or theft) detection and prevention of a MCU is a popular problem with active research, as successfully addressing it would benefit a lot of parties (including the government and the military).

A distinction between ordinary and secure MCUs should be made[11] and, due to the sophistication of the protective mechanisms and the attacks, a broad classification of attackers as[1]:

- **Home Hackers** Clever and curious people with a very limited budget, with some degree of knowledge and no time restrictions.
- **Semi-Professional Crackers** Professionals skilled on electronics with access to specialised equipment and resources. Their time allowance and funding might be limited.
- **Funded Organisations** Organisations with access to MCU manufacturing equipment. Their funding is usually unconstrained and their time schedule is usually tight.

Firmware tampering has a number of consequences, the most obvious being an attacker downloading the code from a MCU and flashing it onto a MCU that they sell, effectively avoiding development and testing costs but still offering the same product as other manufacturers[4]. A less obvious, but perhaps more important, case is the case of back-dooring[1] a MCU by re-flashing on it a modified version of the firmware with coded added by the attacker in order to accomplish their malicious intents. Furthermore MCU firmware might contain government or industrial secrets whose integrity and value is more valuable than the integrity of the MCU.

Current attack technologies and methods will be reviewed (Section 3) and the defensive techniques (Section 4 developed to counter these. These are described to give a (brief) overview of the current state of affairs and, once the ATmega MCU features have been reviewed (Section 2), a working attack against the ATmega will be presented (Section 5). This paper will conclude (Section 6) by summarising the material presented and explaining why securing a device is hard.

2 The AVR MCU Series

The Atmel AVR series is an enhanced-RISC architecture MCU family that consists of the ATtiny, ATmega and ATxmega sub-categories and derivatives of the above, including 32-bit AVRs and application specific FPGAs[2]. The models have varying degrees of hardware capabilities and large operating voltage windows in order to accommodate demand and integrate well with peripherals[3][4].

Developing software for an AVR is easy as the AVRs benefit from the free `avr-libc` high-performance C

1. The act of adding code to a system without the user's knowledge or approval, usually to accomplish nefarious tasks.
2. info:AVR family link
3. info:https://www.newbiehack.com
4. info:http://www.atmel.com/v2PFResults.aspx

run-time library(optimised for the AVR RISC architecture), the `avr-gcc` and `avr-gdb` compiler and debugger(both based on very popular and high quality GNU software tools), the `avrdude` programming software(or Atmel's proprietary `AVRStudio`) and `Simulavr` simulator software. Additionally, Atmel provides proprietary APIs for interacting with the AVR and the developers can choose from a wide variety of programmer units available for working with the AVRs[12].

2.1 ATMega Architecture and Features

2.1.1 Important Feature Overview

The ATmega series of MCUs is a relatively large family of MCUs and the focus of this paper is on the ATmega644 and ATmega1284. The only differences between then 1284 and the 644 is that the 1284 has got more memory available and an hardware extra timer. A summary of (some) of the features of the two units is give in Table 1[5].

Both MCUs are an enhanced-RISC Harvard architecture 8-bit CPU. Figure 2 shows the conceptual difference between a Von Neuman (most modern PCs) and a Harvard architecture, where the key distinction lies in the separation of application code and program data into different memory sections (Harvard) and tasking the CPU with distinguishing between code and data that lives in the same memory region (Von Neuman). The 644/1284 implement a Harvard architecture for both power and computational efficiency, being designed to access more than one registers simultaneously (due to the physical wiring of the CPU), enabling them to execute an instruction per cycle. Their operating voltages can vary between 1.8V and 5.5V (maximum operating frequency 20 MHz).

2.1.2 Memory Organisation

The 644/1284 are equipped with an EEPROM, flash memory, SRAM, a large number of general purpose registers and a large number of I/O registers (in order to be able to perform I/O) and all memory (including I/O memory mapped images) is linear, i.e. it follows the flat memory model.

The flash memory is separated into two regions, the bootloader section and application code section. The boundary between the two sections can be configured by programming the appropriate fuses, and the page size can also be configured that way as well. Both sections hold code, however code residing in the bootloader section can execute a special instruction (`SPM`[5]) which allows the bootloader code to write to *any* section in the flash memory and hence possibly modify itself(designed for purposes such as firmware upgrades). The bootloader code can be triggered by a direct jump from the application section or by programming the reset vector via the reset fuse to point to the appropriate section of the bootloader code.

5. `SPM` = Store Program Code, assembly instruction for the AVR.

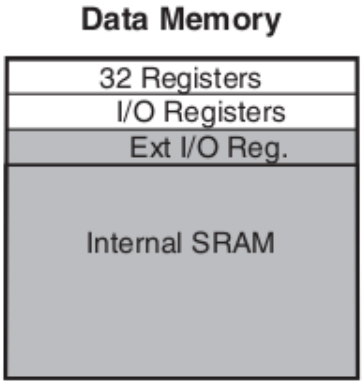


Fig. 1: SRAM layout for the ATmega 644 and 1284. **Reproduced from** [5].

TABLE 1: Memory differences for the AVR ATmega644/1284

Model	EEPROM (Kb)	SRAM (Kb)	Flash (Kb)
ATmega644	2	4	64
ATmega1284	4	16	128

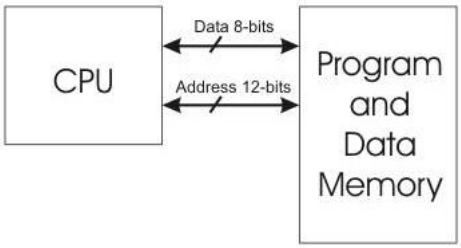
The EEPROM is memory for data that needs to persist between reboots of the MCU and hence it is (widely) used to hold configuration variables and other non-temporary preferences the application code (or the bootloader) may need, having an average lifespan is 100,000 write cycles per page.

The SRAM is volatile storage and is used as the stack and heap for the software (either application code or bootloader code) as well as for storing the Register File (i.e. the 32 GP registers) I/O and Extended I/O Memory. The reserved register locations exist in order to support the use of peripheral units as well as hold program status information (e.g. the Stack Pointer can be found in one of the GP registers).Figure 1 gives an overview of the SRAM hierarchy, which is slightly different (in terms of region sizes) for the 644 and the 1284 as the 1284 offers more SRAM.

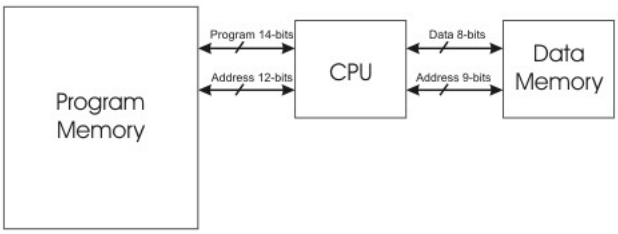
2.2 ATMega Security Features

The AVR ATmega644/1284, even though not meant to be secure hardware modules, posses certain security features. In particular, each board provides six Lock bits responsible for controlling access to the board's memory and prevent reading or modifying the memory (e.g. prevent code executing from the bootloader section to read/write the application code section via the `SPM` instruction). This access control is not permanent, as that would limit the usefulness of the MCU and therefore one has the option to reset the lock bits (i.e. having no protection scheme enabled) by issuing a Chip Erase command, which has the effect of completely erasing the Flash, EEPROM and Lock bits.

The erasing is performed with the sequence of events presented above and this is important, as one does not want to remove the access protection before removing all sensitive data and hence the Lock bits are set to 1



(a) Schematic of a Von Neuman architecture.



(b) Schematic of a Harvard architecture.

Fig. 2: A comparison of different machine architectures. **Source**:[7].

TABLE 2: Security lock bits offered by the ATmega644 and ATmega1284. BLB stands for Boot Lock Bit and LB for Lock Bit.

Lock Bit Byte	Bit Number	Default
BLB12	5	1
BLB11	4	1
BLB02	3	1
BLB01	2	1
LB2	2	1
LB1	1	1

only after the whole program memory has been erased. Even though the flash memory has an average lifespan of 10,000 write cycles (as well as programming being relatively expensive as an operation) this approach makes sense as the ultimate goal is to preserve the intellectual property on the board rather than the board itself.

Table 2 provides an outline of the available Lock bits provided by the ATmega series. The functionality of the BLB1 group is to control access and modification of the bootloader section, group BLB0 bits control access to the application code section and group LB bits are responsible for controlling modifications on the EEPROM and Flash.A detailed explanation of their functionality and how to use them is given in [5].

3 Attacks on Hardware

make this flow with previous paragraphs A distinction between *passive* and *active* attacks should be made. In the former the attacker simply monitors the chip's normal operation and tries to infer the input-output mapping whereas in the latter case the attacker actively manipulates either the chip or its operating environment with the aim of obtaining insight on the chips inner workings.

Attacks on MCUs may attempt to recover a number of artefacts, including cryptographic keys the firmware and do not need to necessarily attack the hardware itself but can exploit flaws in algorithmic design and implementation and protocol failures or inter-component communication patterns[1][9], obtain information by corrupting the memory or exploiting memory remanence[11][6].

The following discussion closely follows [11].

3.1 Non-Invasive Attacks

mention other non-invassive attacks as well and SCA, such as exposure to abnormal temperatures, exposure to radiation or using lasers damage/alter/heat specific portions of the chip. make this more technical. [**?**] Non-invasive attacks are attacks which require no depackaging or special preparation of the chip and hence attacks under this category leave little tamper evidence behind. These attacks might be very time consuming to find and are not guaranteed to be successful, but are very easy and cheap to replicate once found. Furthermore, non-invasive attacks could target badly implemented communication, bugs or security protocols in order to bypass security restrictions.

3.1.1 *Power Analysis*

need sampling equipment with at least double the operating frequency to get a measurement per clock cycle and also check out Correlational Power Analysis [**?**][**?**] . could also include image from DPA paper showing DES rounds. also say set up, how and where you place the resistors. see DPA paper again Different instructions executing on a CPU require different amounts of power also because of CMOS nature 0 takes less power than 1, and hence one can infer which instruction is executing on a CPU by analysing a power trace generated by the MCU. These attacks are easy and relatively inexpensive to perform as they only require widely available tools.

Simple Power Analysis(SPA) involves direct observation of the MCU when it performs cryptographic operations and can leak information about both the keys and the cryptographic operations themselves (i.e. nature or structure of the algorithm)[9][2].

Differential Power Analysis(DPA) extracts sensitive information by using statistical techniques on very large traces. The techniques involves obtaining power traces of known cipher-texts(but not necessarily knowing the corresponding plain-texts) and individual bits of the key are recovered by analysing the differences in power consumption[9][2].

One can generally avoid noise in their power measurements by sampling the voltage (usually) on the ground line[11].

3.1.2 *Glitch Attacks*

say that glitch is inserted somewhere and that somewhere can be shown tracked down by power trace analysis. trial and error, repetitive process and can systematically be explored [**?**] *Power glitches* and *clock glitches* aim to make the CPU skip or execute incorrect instructions by applying transients. This attack can target in individual components of an MCU and a systematic search can deduce which components are affected by a given glitch sequence.

Clock glitches involve increasing the clock signal frequency so that some flip flops sample their input before being updated and hence report an incorrect value. Clock glitches are mainly aimed against software-based protection mechanisms, affecting CPU operation by supplying the CPU with incorrect data.

Power glitches work by supplying either too much power or too little, shifting transistors' threshold and causing flip-flops to read their state incorrectly. Power glitches need to be carefully synchronised with the internal clock and prolonged attacks might damage the board.

Glitch attacks are especially dangerous as they may abuse the program counter in order to map out the memory, which is linear as described in 2.

3.1.3 Data Remanence

explain more, make more technical Prolonged exposure of SRAM cells to the same values can make the cells 'remember' their state, due to material properties and stress[6]. Furthermore, cooling the memory down can prolong the time for the data to leave the memory [6][10][11]. If, for example, a start-up routine always writes security keys to the same memory location, after some time they key will be recoverable by looking at the physical state of the memory or by cooling down the chip, starting it and then reading the memory.

EEPROM suffers as well, but to a lesser extent, as material-wise one can only tell virgin-cells from used cells[11].

3.1.4 Timing Attacks

Timing attacks exploit the software implementation of cryptographic algorithms. Compiler optimisations (avoiding unnecessary branches, register and cache usage) and other implementation choices make the execution time of an algorithm dependent on the input and the secret key, rather being fixed for any input. For example, when input is compared byte-wise with a key and rejected when the first non-matching byte is found, rather than first consuming the whole input string.

Different instructions take different time to execute(e.g. `MOV eax,[eax]` is considerably slower than `INC eax`) and thus one could collect timing information for various input messages and systematically deduce the correct key.

If timing information is correlated with power analysis then defences such as constant instruction execution time could be defeated. One might use `NOP`s in the case of a wrong key in order for rejection and confirmation responses to have constant execution time but `NOP` consumes substantially less power than `INC eax` and correlating timing and power consumption information would reveal this.

3.2 Semi-Invasive Attacks

Semi-invasive attacks require depackaging of the chip but do not destroy the passivation layer as no electrical contact with the chip is needed. Semi-invasive attacks can be automated and can yield results faster and cheaper than invasive attacks.

3.2.1 UV Light Exposure

Older chips and chips that are designed to withstand low-cost non-invasive (i.e. no depackaging) attacks are susceptible to having parts of the memory altered if it is exposed under UV light. Security fuses that prevent read-back of the memory could have their state reset by sufficient exposure under UV light. The attacker must locate the security fuse though, which can be very tricky.

3.2.2 Backside Imaging

Backside imaging involves shining IR light on the rear side of the chip and imaging it from this angle, since it is a mirror-image of the front side. This is possible because, usually, light shown through the backside does not have to go though multiple layers and hence protective metal meshes(discussed in Section 4) or normal chip layers are avoided. On some chips it is possible to extract ROM contents via this technique.

3.2.3 Photo Carrier Stimulation

Optical Beam Induced Current and Light Induced Voltage Alteration are the two most common techniques for failure analysis that take advantage of the photoelectric effect. These techniques involve shining lasers on the semiconductor surface in order to alter some property; in OBIC a slight current is created and by analysing this one can deduce the device's properties (including defects and anomalies) and produce an image of the the board being scanned, while in LIVA the board is connected to constant power supply and changes in the power supply are monitored as laser is shone on the device, allowing one to deduce the device's characteristics and construct an image[3].

Similar to Backside Imaging, lasers can be used to read the state of memory cells in CMOS SRAM.

3.3 Invasive Attacks

Invasive attacks require direct access to the board's surface and as a result destroy the packaging in the process, therefore leaving tamper evidence. Invasive attacks usually aim to understand how a MCU works and then develop cheaper attacks for that chip, as invasive attacks are laborious, require expensive equipment and highly skilled attackers. look at lecture 1 of HWRevEng course

3.3.0.1 Exposing the chip surface: This usually involves destroying the packaging by using chemicals or drilling(or other methods). While this is a process that is not very complicated[11], one might have trouble finding the chemicals required. An alternative for depackaging the chip is to send it to a failure analysis lab[8].

3.3.0.2 Reverse engineering: both hardware and the software. Hardware reverse engineering requires using reflected light microscopes or SEM[6] for constructing a complete image of the surface. Layer removal might be required if deeper layers are not visible in order to have a complete view of the device. Software reverse engineering can be accomplished when one has obtained access to the memory.

3.3.0.3 Micro-probing and Modification: In microprobing sub-micrometer thickness probes are used to establish contact with the bus lines in order to observe and manipulate bus signals. To achieve reliable results a micro-probing workstation[7] is used, consisting of a microscope, micro-positioners for the probes, a movable base and a test socket to place the chip. In order to establish contact with the bus lines the passivation layer should be removed, usually done with UV or green lasers, and access to bus lines of deeper layers can be achieved by using a FIB workstation.

Modifications to the MCU's components (adding new interconnects or destroying circuits) are not always necessary, but could prove useful. For chip modifications to be successful, the attacker must be sophisticated and must have at least partially reverse-engineered the board.

4 Countermeasures to known attacks

4.1 Physical Protection

asic glue logic design, protective metal-mesh layer, shrink things, mesh of wires, encase in epoxy, make layers destroy each other if removed. doping can also reduce light penetration to reduce imaging attacks

4.1.1 Protection circuits

radiation/temperature/voltage/frequency detector circuits that cause reset on abnormality detection (instability reference [1]). add transistors on top to hide true signal (provide reference) keep keys in own, self powered module.

4.2 Side-channel Protection

decrease signal/noise ratio (either by introducing noise or making the signal smaller), constant time/power operatations, insert random delays[11], [9] and shielding the device.

Planarization defeats optical inspection with microscope (source: Introduction to hardware security and trust, sergei skorobogatov paper).

- overview of most popular techniques
- benefits and how they improve the situation/approach the problem
- added cost for this investment (in terms of hardware and money, transparency to the developers, runtime overhead etc)

perhaps review some popular secure chips ?? IBM 4758 is a secure device [8], some Dallas chips and perhaps more.

thermally enhanced packaging : makes difficult to microprobe chips as they need to be cooled or they'll fry [slides of hardware reveng module] and careful decaping needed

ceramic packaging (Al2O3 is common, often mixed with SiO2, AlN and BeO) is extremely chemical resistant and seals very good, but can crack

potting : protects board from environment[water/dirt] but might also make tampering harder and may need to be removed conformal coating : same deal as potting, but it's a thin layer that doesn't fill the enclosure but just covers the top. underfill : provides stability but gets in the way as well(makes desoldering harder).

5 Attacking The ATmega

The ATmega is a typical MCU that is susceptible to a successful attack by the **Home-Hacker** class of attackers due to its weak protective mechanisms. While there is a way to deliver firmware updates to the MCU securely[4], keeping the firmware secure on the device is impossible.

The motivation behind attacking the ATmega 688/1284 is their popularity and to prove how easy it is

5.1 Motivation

5.2 Attack Overview

* added cost (in terms of $$, extra hardware and software implementation penalties/overhead)

6 Evaluation

mention legal issues : HWRevEng No system is unbreakable and one can only harden their system enough to make the effort of breaking it unbearable to those they wish to protect against[1][11].

security is hard to do because one must carefully analyse and model all threats [9]

6.1 Attacks and Solutions overview

present how expensive it would be for someone to put together a probing station and in general give estimations of cost and skillset required for various attack categories

6. SEM = Scanning Electron Microscope
7. all moving components should have micrometer precision
8. http://www.cl.cam.ac.uk/ rnc1/descrack/ibm4758.html

6.2 Conclusions

REFERENCES

[1] Ross Anderson and Markus Kuhn. Tamper resistance: A cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce - Volume 2*, pages 1–11, Oakland, California, November 1996. USENIX.

[2] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In Bruce Christianson, Bruno Crispo, Mark Lomas, and Michael Roe, editors, *Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer Berlin Heidelberg, 1998.

[3] Jr. Cole, E.I. Beam-based defect localization techniques. In *Microelectronics Failure Analysis*, Materials Park, Ohio, October 20011. ASM International.

[4] Atmel Corporation. Atmel AVR231: AES Bootloader. Application Note 2589E-AVR-03/12, Atmel Corporation, 2012. www.atmel.com/Images/doc2589.pd.

[5] Atmel Corporation. ATmega164PA/324PA/644PA/1284P Datasheet. Datasheet 8272E-AVR-04/2013, Atmel Corporation, 2013. www.atmel.com/Images/8152s.pdf.

[6] Peter Gutmann. Data remanence in semiconductor devices. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM'01, pages 4–4, Berkeley, CA, USA, 2001. USENIX Association.

[7] Philipp Hof. A Primer To Microcontrollers. http://www.elec.canterbury.ac.nz/PublicArea/Staff/hof/p10-embed/p10-tutorial/index.html, March 2014.

[8] Andrew Huang. Hacking the PIC 18F1320. http://www.bunniestudios.com/blog/?page_id=40, March 2014.

[9] 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.

[10] Sergei Skorobogatov. Low temperature data remanence in static RAM. Technical Report UCAM-CL-TR-536, University of Cambridge, Computer Laboratory, June 2002.

[11] 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.

[12] Alan Trevennor. *Practical AVR microcontrollers games, gadgets, and home automation with the microcontroller used in Arduino*. Apress, Berkeley, CA New York, 2012.