On the Impact of Automating the IC Analysis Process

Olivier THOMAS
Texplained SARL
olivier@texplained.com

Dmitry Nedospasov
Technische Universität Berlin
dmitry@sec.t-labs.tu-berlin.de

Abstract

In the security analysis of hardware, invasive analysis techniques are of particular interest because they are the only class of attacks capable of defeating all known embedded and software countermeasures. However, invasive analysis techniques are often disregarded because it is generally assumed that there is a substantial amount of effort required to perform these attacks. This work introduces and evaluates a new approach for the security analysis of Integrated Circuits (ICs). These techniques are based on years of development and experience resulting in a suite of tools known as ARES (Automated Reverse Engineering Software). Through substantial automation of the IC analysis process the amount of effort required can be significantly reduced. Additionally, this work presents practical results for a modern secure microcontroller. With ARES it was possible to identify security-relevant signals across the device and develop strategies to extract all the memories of the target device. The amount of time required to conduct the study was reduced from several months to just a few days. Moreover, ARES was capable of reproducing the device's design files, making it possible to outsource parts of the extraction to a professional lab at a low hourly rate.

1 Introduction

The security threats faced in the field by hardware are very different from those faced by software. While software can be readily patched, hardware generally requires an entirely new revision of the device. As with other areas of security, hardware security and Integrated Circuit (IC) security in particular have been defined by threats that these devices face in the wild. The industries most famous for facing hardware level exploitation is PayTV [4]. Since a broadcast medium is used to deliver content there is no communication back to broadcaster. Hence, the security of the broadcaster's system is the security of the individual subscriber cards as the broadcaster has no real way to detect clones that are in the field. Similarly counterfeit peripherals, such as game controllers and printer cartridges, threaten the profits of a manufacturer selling genuine peripherals. In all of these cases, the IC is a hardware trust anchor, storing the proprietary algorithms and cryptographic secrets used for authentication. Hence, a device clone that is able to emulate the behavior of a genuine device is indistinguishable from the original.

Hardware security analysis techniques are generally classified into three categories depending on the amount of sample preparation required. As a result these classes of analysis techniques also vary in the amount of effort required to perform analysis. Depending on the target, gauging the amount of effort that is actually required to mount a successful attack in practice can be very difficult. The actual cost of the attack can be amortized across many device instances, making the per device cost of the attack relatively low, as is the case with counterfeit devices. Moreover, the amount of effort required to perform the attack can also greatly be reduced through the automation of the analysis process.

Academic researchers as well as the security certification bodies tend to focus on so-called *non-invasive* and *semi-invasive* analysis techniques. Non-invasive techniques are particularly attractive as they require no sample preparation and very little equipment beyond standard test and measurement equipment that is readily available in every university electronics lab. Non-invasive attacks include side-channel analysis techniques, as well as fault injection techniques, such as clock and voltage glitching [3, 7]. For non-invasive analysis it is also entirely possible to instrument testing during development to make any practical non-invasive attacks infeasible. While non-invasive attacks focus on the security of the entire system, semi-invasive analysis techniques are substantially more powerful as they offer a degree of spatial selectivity once the device is opened.

This makes it possible to target only particular areas of the device, which makes it possible to mitigate certain system-level countermeasures that prevent non-invasive attacks. Nevertheless, semi-invasive attacks also require synchronization with the target, which can be difficult to realize in practice [14].

Certification standards such as Common Criteria define a considerable suite of non- and semi-invasive tests that must be performed during the certification process. As a result, new devices that receive one of the highest Evaluation Assurance Levels (EALs) have undergone extensive testing against non- and semi-invasive attacks before entering the field. The situation is very different for fully-invasive attacks. Fully-invasive attacks are considered to be attacks that physically manipulate the underlying circuit and are particularly difficult to prevent. These attacks range from attacks where data is manipulated on during runtime on a device's internal data bus, to permanent physical modification of the circuit [5, 6]. Because fully-invasive analysis techniques target the underlying circuit, an understanding of the underlying circuit is required. For this reason Common Criteria defines requirements for trusted parties during manufacturing, as well as additional requirements pertaining to the confidentiality of designs and design files.

Nevertheless, it is possible to perform black-box fully-invasive analysis of secure devices as well. Generally, the first step in this case is to gain an understanding about the underlying design. For example, if the device utilizes an unencrypted Read Only Memory (ROM) then the firmware of the device can be directly extracted from optical images of the ROM. For this reason manufacturers have shifted to storing secrets such as device keys in other non-volatile memories such as EEPROM and Flash, requiring attackers to extract secrets at runtime. Without access to documentation identifying areas of the device with access to this additional information requires the attacker to partially or fully reverse-engineer the IC. However, once elements of the CPU architecture have been found, the attacker is capable of mounting a successful attack by modifying the control flow of the CPU by manipulating the data directly on the data bus [6]. With the firmware and the contents of the non-volatile memories it is feasible to produce a counterfeit device capable of emulating the original.

To prevent such attacks outright, manufacturers began to introduce something known as Custom Hardware Functions (CHFs). CHFs implement parts of the manufacturer's proprietary algorithms as a logical circuit at the transistor level versus at a hardware-level alone. CHFs are commonly used to implement cryptographic algorithms as coprocessors so that the device firmware no longer must contain any proprietary data. An attacker faced with such a device is then forced to fully reverse-engineer the cryptographic circuit at a transistor level [8]. Moreover, modern process geometries are making such attacks increasingly inefficient and infeasible in practice. Optical imaging of the circuit is no longer sufficient to reliably resolve individual signals on the device. With insufficient resolution it is impossible to fully resolve the individual gates, but also near impossible to reliably trace signals through the circuit [12].

This work introduces the techniques necessary for reliably analyzing modern security ICs. The techniques are not limited by the geometries of modern processes. Furthermore, they are capable of reliably reconstructing the underlying logical circuit, making it possible to identify, analyze and emulate circuits containing CHFs.

The main contributions of this work are the following:

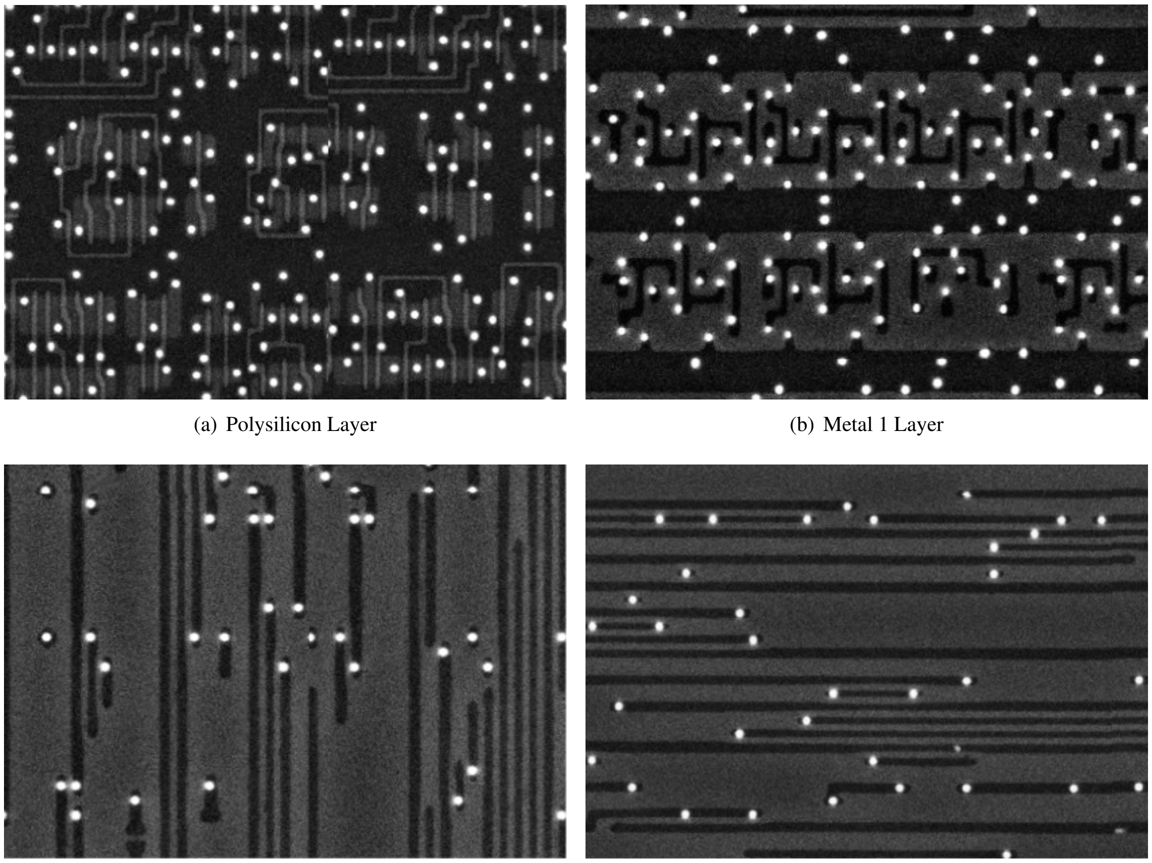
1. The deprocessing and imaging techniques necessary for automated extraction of modern chip features. Based on approximately 10 years of designing automated IC analysis tools, we present the formula that we have found to be the most effective.

2. The ARES (Automated Reverse Engineering Software) analysis suite. ARES is a suite of tools that were developed over the years to perform IC analysis in an automated fashion. Although ARES was used to analyze the target device, any software with the capabilities of ARES should be able to produce similar results. Specifically, ARES is capable of extracting all the relevant features of the circuit, reconstructing the netlist of the device and tracing security-relevant signals in automated manner.

3. A study of a secure microcontroller that remains undefeated in the field. This microcontroller implements many modern countermeasures against analysis. This device employs additional protection layers (shields), utilizes encrypted embedded memories and includes CHFs in the design. With ARES we were able to identify the CPU architecture of the target, identify the instruction registers for extraction of the device's flash and identify the data buses used for SRAM and ROM as well as the encryption functions. ARES was also capable of producing a GDS2 file, which would make it possible to outsource the process of invasive microprobing to a professional lab at a low hourly rate.

4. A comparison against state-of-the-art techniques. We provide an overview of the amount of effort that would be required to develop ARES as well as how much effort is required to analyze a target.

The rest of this work is structured as follows: Section 2 presents prior research in the area of security analysis of ICs. Next, the requirements for deprocessing and


(a) Polysilicon Layer (b) Metal 1 Layer

(c) Metal 2 Layer (d) Metal 3 Layer

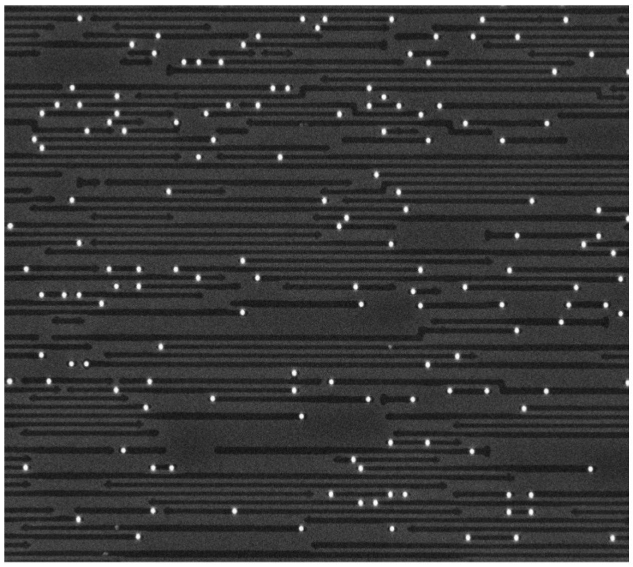
Figure 1: SEM Images of the different layers of the IC stack. The material contrast within the images allows for reconstruction of the circuit in an automated fashion. The contacts (white), polysilicon transistor gates (grey), as well as the dopant regions (darker grey), can be clearly differentiated from the silicon substrate (black) in the image of the polysilicon layer, see Figure 1(a). The vias going down to the polysilcion layer (white) and the removed metal lines (black) can be clearly differentiated form the surrounding passivation (dark grey) in the image of the metal 1 layer, see Figure 1(b). The removed metal interconnects (black), vias (white) can be clearly differentiated from the passivation on the interconnect layers, see Figures 1(c) and 1(d).

an introduction to the ARES analysis suite as well as the workflow are presented in Section 3. Section 4 presents a study of a secure microcontroller that implements many modern countermeasures against invasive analysis. Finally, a comparison in terms of effort as well as the implications of this research are presented in Section 5.

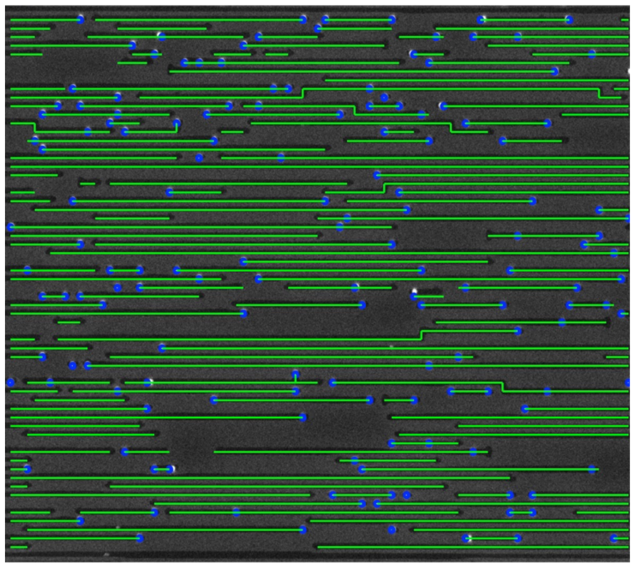
2 Related Work

As semiconductor technologies evolve, so do the analysis techniques. The Failure Analysis (FA) industry is responsible for coping with the challenges posed by the production of the latest generation of semiconductor devices. Failure analysis equipment is developed in conjunction with the latest generation of process technologies to make post-production testing and verification possible. Similarly, the actual techniques used in failure analysis were developed as a means of identifying and diagnosing manufacturing errors in new processes [10].

Security analysis techniques for ICs have evolved over the years and can be adapted for modern processes. For invasive analysis destructive delayering and subsequent imaging of the device layers is generally necessary. These steps help an attacker to understand the underlying circuitry, identify the memories and understand the overall architecture. If the firmware is within the ROM and the ROM is unencrypted, the firmware can be extracted directly from the optical images of the device [6]. However, if the secrets are embedded in another memory or if the cryptographic algorithm is implemented in the form of a CHF within the device's logic, then the attacker is forced to study the optical scans of the IC. In



(a) Source Image



(b) Detected Interconnects

Figure 2: SEM image of the Metal 3 layer that has been corrected for distortion, see Figure 2(a). The features detected by the ARES analysis suite are visible in the image with the extracted features, see Figure 2(b).

the case of invasive microprobing, identifying the data path from the program memory to the instruction registers within the CPU core is sufficient. However, with proprietary cryptographic functions, reverse-engineering and reconstructing the logical function from the logical gates is necessary [8].

Automated reverse-engineering tools have been introduced in the past, the most prominent of which is degate [1]. Most works to date utilize optical images for performing destructive delayering and imaging of ICs. However, the smallest geometries within any process are the transistor gates of the cells. Hence, such tools are not capable of reliably resolve the cells of the design. Optical images also lack the material contrast, which means that such tools are very resource intensive. The material contrast of SEM images makes it possible to reconstruct the device as a set of features instead of a pixel matrix, see Section 3.

One of the most important techniques for fully-invasive microprobing is known as memory linearization or Linear Code Extraction (LCE). In this technique the attacker manipulates the control flow of the execution by manipulating the opcodes directly on the device's data bus. If branching is prevented, the microcontroller's program counter continues to increment the address linearly, allowing the attacker to observe the entire contents of the device's firmware on the data bus [2]. The signals on the data bus can be set to static values by using a Focused Ion Beam (FIB) to connect them to one of the supply rails (VDD or GND). Alternatively the control flow can be manipulated dynamically, by driving the signal on the bus using a custom probing needle buffer/driver. Most importantly, since embedded memories are encrypted on modern secure microcontrollers, an attacker who succeeds in isolating the instruction registers will be able to manipulate the control flow even if the data is encrypted [9, 12]. Because CPUs are incapable of executing encrypted data, the data must first be deciphered before it reaches the instruction registers [5].

3 Approach

This section describes the criteria that are necessary for producing SEM images for automated feature detection. Additionally this describes the ARES analysis suite that is used in the analysis of the target in Section 4. ARES was developed over the years to determine what is possible with automated analysis. However, any software with the capabilities of ARES should produce similar results.

3.1 Sample Preparation and SEM Imaging

For the reliable automation of the IC analysis process, high-quality images of the IC are required. To date, the tool of choice has been optical imagery using confocal microscopy [6]. Confocal images have the advantage of preserving depth information and making it possible to observe multiple layers simultaneously. However, multiple visible layers also make automation more difficult. Hence, the technology of choice for imaging modern

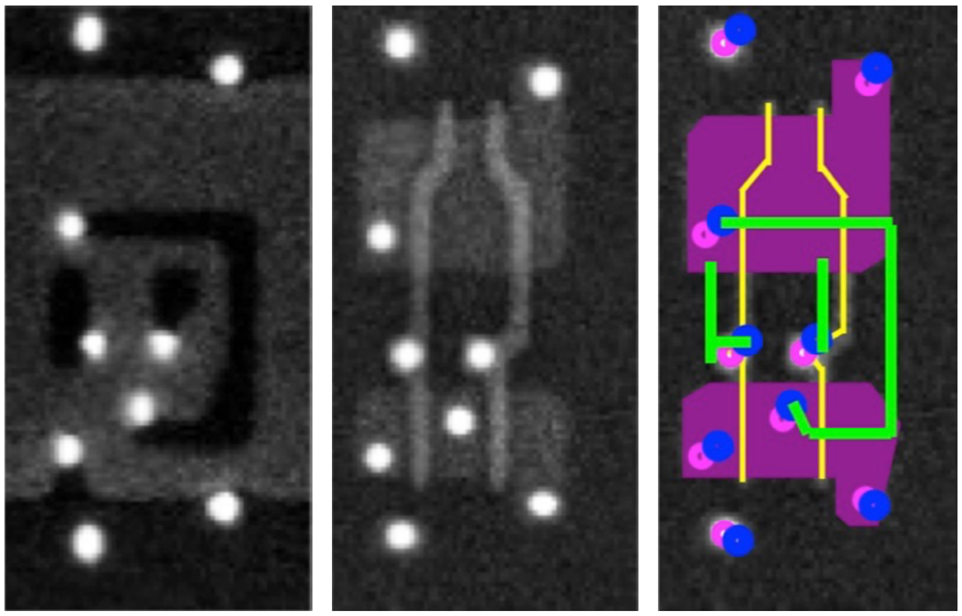

(a) Source Image (b) Detected Interconnects (c) Detected Interconnects

Figure 3: SEM images of the polysilicon and metal 1 layer of a NOR gate, see Figures 3(a) and 3(b). Through a semi-automated process, the entire cell library of a device can be reconstructed with ARES. The resulting representation of the gate within the database of ARES is also presented, see Figure 3(c).

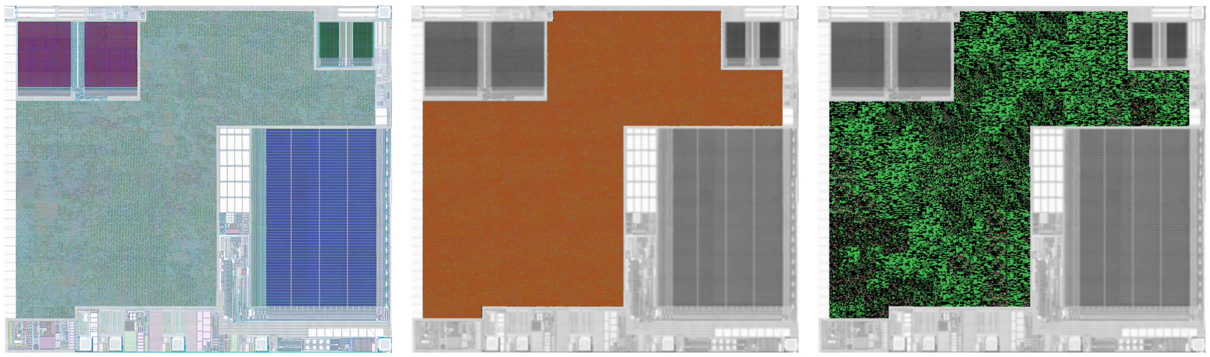
planerized processes with small geometries are Scanning Electron Microscopes (SEMs).

A SEM works by scanning an electron beam across the device's surface and measuring the current detected by backscatter electron detectors. SEMs are commonly used in material sciences as both topography and material contrast are visible in the resulting images. This also implies that SEMs are capable of imaging differences in the dopant layers of the IC [11]. In order to produce SEM images that can be reliably detected in an automated fashion, particular care has to be taken when preparing the samples for imaging. Sample preparation is a mix of mechanical polishing, Reactive Ion Etching (RIE), and wet chemicals are used to prepare the sample for the SEM. With modern technology nodes, optical imaging of cell library is no longer possible. However, the resolution of a SEM is sufficient for imaging any CMOS process.

SEM images have inherent distortion, and software must be used to correct the individual images before reconstructing the scan of the full chip. For the results presented in this work, correcting the distortion within the individual images was handled by ARES, see Section 3.2. Modern secure microcontrollers have 6 metal layers or more depending on the design of their additional protective layers known as shields. Each metal layer must imaged individually at a resolution higher than that, which is possible with optical images. The three key components of the circuit, i.e., the signal traces (metal wires), interconnect vias (metal) and passivation (oxide) of the circuit are all clearly visible and easily distinguishable from one another based on their contrast in the resulting SEM images, see Figure 1.

3.2 The ARES Analysis Suite

Throughout this work the proprietary ARES analysis suite was used for analysis. ARES is based on the lessons learned from many years of development of automated tools. The ARES analysis suite automates the analysis process and performs many common tasks without user interaction. Although we chose to focus on the tool that we develop and use professionally, any tool with the capabilities of ARES should be able to produce similar results. For an automated IC analysis tool, the feature extraction process can be summarized as: The input format for the software are SEM images of all relevant layers of the device. Because of the inherent distortion of SEM images, the software first transforms and corrects these images for distortion. Next, the transformed images are correlated and chip is reconstructed layer by layer. Subsequently, features are detected within every layer. For this, the material contrast and topography data that can be extracted from the contrast of the SEM images is used to identify the three main components on the interconnect layers, i.e., metal lines, vias and passivation, see



(a) Optical die shot (b) Extracted interconnect layers (c) Detected cells

Figure 4: Optical die shot of the target secure microcontroller, as well as overlays for the interconnect layers and the detected standard cells. The target device contains a ROM (purple, top left), an SRAM (green, top right) as well as a flash memory (blue, right), see Figure 4(a). From the reconstructed interconnect layers the buses and corresponding encryption functions could be identified, see Figure 4(b). As interconnects are followed and reach the standard cells of the gate, the standard cells are reconstructed and added to the ARES database. The detected standard cells and their location on the device are also clearly visible, see Figure 4(c). The overlays are the result of approximately one week of analysis using the current version of the ARES suite.

Figure 2. Because ARES operates on the detected chip features, it is not as resource intensive as tools that operate on the pixels of the optical images, such as [1]. This also eliminates the necessity to stitch images prior to analysis. Following this step, each of the interconnect layers can be connected to one another within the internal model of ARES. Finally, the first interconnect layer, metal 1, is correlated and aligned with the underlying polysilicon layer.

At this stage the chip is fully aligned and the design of the IC can be reconstructed within the analysis software. Additionally, ARES is capable of automatically detecting, producing and outputting the design files of any interconnect layer within the IC stack, see Section 4. To reconstruct the cell library, a semi-automated process is employed in the current generation of ARES, although this can be automated in the future, see Figure 3. Standard cells are provided as library by the foundry and are not automatically generated during synthesis. For this reason, standard cells are currently traced manually within ARES, after which all gate instances of the cell across the device can be detected automatically. The typical size of a standard cell library for a modern secure microcontroller ranges from approximately 60-100 standard cells.

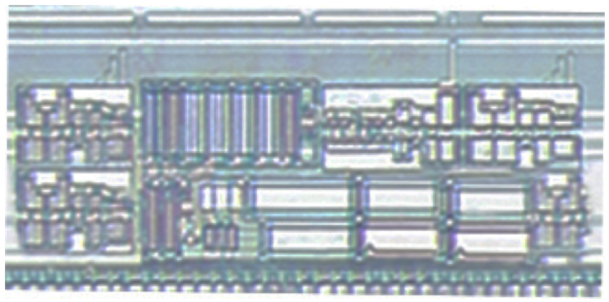
The typical workflow for security analysis involves tracing signals across the device and within the target's CPU core. This requires first identifying a signal of interest, for example, an output of a memory. Next, this signal can be followed in automated fashion until a standard cell is reached. If the cell instance is part of the reconstructed standard cell library, it is automatically displayed. If not, then the user can choose to reconstruct the cell and add it to the library or simply mark it and follow one of its inputs or outputs. During the entire process ARES visualizes a netlist representation including the path followed as well as the interconnects. Rather than simply following the outputs, following additional inputs, such as select lines of multiplexers, can provide important hierarchical information about the architecture, see Section 4.

4 Results

This section presents the results of the analysis of a secure microcontroller, which remains undefeated in the field for over 10 years, see Figure 4. ARES was used to perform the analysis of the target device. However, any other suite with the capabilities of ARES should be able to produce similar results. The secure microcontroller analyzed in this work utilizes many modern countermeasures and obfuscation techniques. To protect the manufacturer, the exact make and model of the device will not be disclosed in this work. These include encryption of embedded memories as well as embedded CHFs. The CPU architecture was not known prior to analysis, but could be determined by the cell library, which was shared with general-purpose mircrcontrollers. Specifics related to the instruction set also gave clues about the overall CPU architecture.

4.1 Target Design and Architecture

A common goal of IC analysis is to extract the device firmware so that the target device can be emulated. Em-


(a) Single flash buffer.

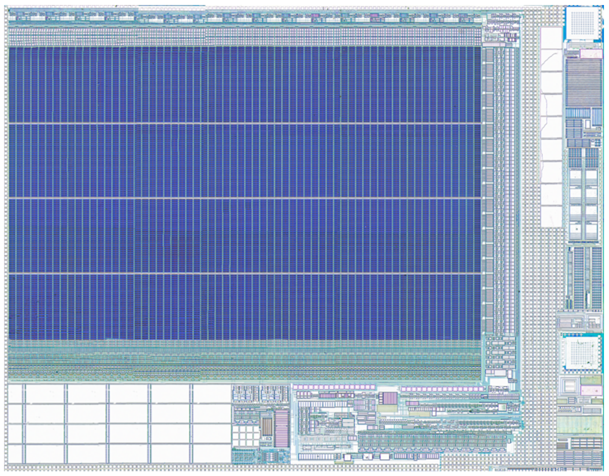

(b) Layout of the flash memory.

Figure 5: The layout of the flash memory of the target device. The 16 output buffers of the flash memory are clearly visible above the flash, see Figure 5(b). The layout of each individual output buffer is identical, see Figure 5(a).

ulation implies either software emulation on a different platform or a device clone in a similar form-factor. For the extraction of the firmware and any relevant secrets stored on the device, the CPU architecture of the device must first be identified. Subsequently, strategies can be developed for the extraction of the embedded memories. The target device contained the following memories:

Two SRAM banks. This type of volatile memory is used to store values at runtime. These memories can contain relevant information for entry points to any bootloaders implemented in the device's ROM. Additionally these memories may contain data that has been deciphered by any CHFs on the device.

Flash memory. This type of memory contains the program memory of the secure microcontroller. Although these memories can be distracted directly, using techniques such as Atomic Force Microscopy (AFM), direct extraction of the flash is a time-consuming process [13]. The most reliable way to extract the program memory are dynamic microprobing techniques at runtime, such as LCE.

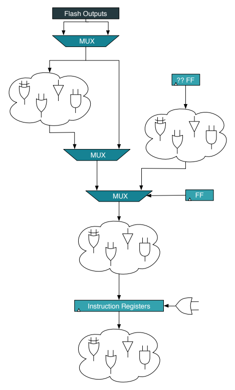
Read-Only Memory. Such memories can have scrambled addressing schemes and contain data encrypted with a CHF as well as other obfuscation techniques. The encryption algorithms are often proprietary and undocumented. Nevertheless, since the ROM is manufactured as a physical circuit it can be extracted without powering up the device. To decipher such memories the encryption must be studied. To further complicate analysis the encryption function may be synthesized within the CPU core.

4.2 Tracing the Flash Outputs

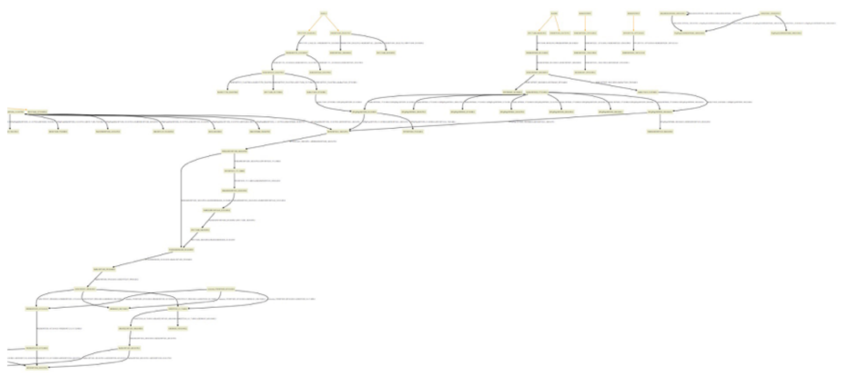
Based on the size of the memories it was safe to assume that the program code was stored in the flash memory. The 16 outputs buffers of the flash memory were quickly identified along the perimeter of the flash, see Figure 5. Subsequently, a single flash output was chosen as a starting point and traced into the core. Each gate that was identified along the data path was analyzed, reconstructed and added to the ARES database. Any subsequent instances of this type of gate are then automatically identified across the device and any additional instances along the data path are automatically added to the schematic.

One type of gate that is of particular interest for analysis are the multiplexers. Since multiplexers are commonly used to arbiter signals on the data bus, multiplexers on a common bus share control signals. Hence, by identifying one multiplexer any additional multiplexers at this stage can be identified by their shared control signals. The control signals of every multiplexer found on the data path were followed, identifying all the multiplexers at every stage and yielding the width of the data on the bus. In this fashion it was possible to quickly identify the width of the instruction set, which was not known prior to analysis.

By continuing to follow the data path, three sets of multiplexers could be identified before reaching the first set of registers within the core. Following the control signals of a single register confirmed that it was shared



(a) Data path schematic.



(b) ARES graph tracing.

Figure 6: By following the flash outputs into the core it was possible to identify the instruction registers of the device. Multiple paths existed through the circuit, however by following the outputs of the multiplexers a single set of registers sharing a common clock as well as control signals could be identified, see Figure 6(a). The graphical visualization of the tracing engine of the ARES analysis suit is also shown for comparison, see Figure 6(b).

with additional registers corresponding to the bus width through the multiplexers. At this stage it would be safe to assume that the identified registers are the instruction registers of the device. With a FIB workstation it would be possible to quickly verify this assumption by exposing the data bus and/or driving values on the bus. However, to experiment with the capabilities of the ARES analysis suite, the outputs of the registers were followed instead.

Because the number of possible CPU architectures could be drastically narrowed down simply by the instruction width, it made sense to look deeper into the core. Unlike with the input of the instruction registers, the outputs of the instruction registers did not follow a single path. Following the instruction registers are instruction decoders, ALU as well as the logic for implementing all other operations with the CPU architecture. By following the outputs of the instruction registers, a set of multiplexers could be identified whose width corresponded to the maximum size of a relative jump instruction within the assumed CPU architecture. Additionally a set of latches was identified that corresponded to the data width of the CPU architecture. Overall it took approximately one day to isolate the instruction registers with ARES, a process that would normally take a month of work or more. Without ARES it would be infeasible to manually study and trace the outputs of the instruction registers in a reliable manner, see Figure 6.

4.3 Tracing the ROM and SRAM

The SRAM and ROM were also studied to identify whether or not these memories were encrypted. Approximately a day was allotted to the study of each one of these memories. A single output of the ROM was initially followed and XOR gates could be identified along the data path. On a data path, XOR gates generally signify that the data path contains a decryption function. The width of the data path could be determined from the amount of ROM outputs, the amount of XORs at each stage as well as the multiplexers at the output decryption block. Additional registers could also be identified along this path. This implies that a state register may be necessary for the decryption meaning that the data is not deciphered in a single clock cycle. Although a full study of the ROM encryption was not completed, parts of the encryption were identified. A full study of the encryption function would make it possible to extract the contents of the ROM directly with ARES from the reconstructed images of the ROM.

The target device contained two SRAM banks. The outputs of the SRAM were traced into the core in a similar fashion to the ROM and flash. As expected, the width of the data path of the SRAM corresponded to the data width of the CPU architecture. A CHF could also be identified along the data path. The presence of an encryption function for the SRAM may imply that the SRAM is shared with a CHF implemented on the device. However, due to time constraints, this was not analyzed any further.

5 Conclusion

To gauge the effectiveness of automated IC analysis, it is important to understand the amount of effort that would be required to reproduce the results presented in this work. With an automated analysis tool, approximately two weeks would be required for deprocessing, imaging as well as verification of the extracted features. After these steps, automated analysis can begin with a much higher degree of confidence than manual tracing. Overall, the automated analysis drastically the analysis times of ICs. Within one week, 22,000 gate instances within the core had been identified and detected with ARES. Tracing the flash and identifying the instruction registers took approximately one day. Several days were allotted for verifying that the logic that followed the instruction registers corresponded to the assumed instruction set of the target. Tracing the SRAM and ROM took approximately 2 days to identify the data path into the core. It is difficult to gauge how long a full study of the encryption functions would take, but it is safe to assume that it should not take more than a month of effort based on the overall complexity of the circuit. Most importantly, extraction of such a cryptographic function, synthesized within the core, would not otherwise be possible without an automated IC analysis tool such as ARES. With automated analysis tools, identifying extraction points within a design becomes trivial. Should such tools ever become readily available, a significant increase in piracy can be expected.

Future Work

The ARES analysis suite also introduces several possibilities that have yet to be investigated. Since ARES is capable of generating the GDS2 of the analyzed circuit, any invasive FIB edits could be outsourced to a professional lab. Since ARES identifies instances of a particular gate type across the entire device, it would be possible to use ARES in conjunction with semi-invasive analysis techniques. For example, it would be possible to perform semi-invasive laser attacks just on the registers of the device, targeting specific elements within the register cell. This would dramatically improve scan times by eliminating the necessity to scan across the entire device. Finally, ARES provides a basis for performing analysis of CHFs, something that was previously considered to be infeasible. In the time required to identify the instruction registers, automated IC analysis software could instead be used to fully analyze and extract a CHF. Another important research direction is to develop design rules and technologies for protecting ICs even if automated IC analysis is possible.

References

[1] Reverse engineering integrated circuits with degate. `http://www.degate.org`. Accessed: 2015-07-24.

[2] ANDERSON, R. J. Security Engineering. In *A Guide to Building Dependable Distributed Systems*. Wiley, Nov. 2010, pp. 483–521.

[3] BAR-EL, H., CHOUKRI, H., NACCACHE, D., TUNSTALL, M., AND WHELAN, C. The Sorcerer's Apprentice Guide to Fault Attacks. In *Proceedings of the IEEE* (2006), pp. 370–382.

[4] CHENOWETH, N. *Murdoch's Pirates*. Before the phone hacking, there was Rupert's pay-TV skullduggery. Allen & Unwin, Nov. 2012.

[5] HELFMEIER, C., NEDOSPASOV, D., TARNOVSKY, C., KRISSLER, J. S., BOIT, C., AND SEIFERT, J.-P. Breaking and entering through the silicon. In *CCS '13: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (Nov. 2013), ACM Request Permissions.

[6] KÖMMERLING, O., AND KUHN, M. Design Principles for Tamper-Resistant Security Processors. *USENIX Workshop on Smartcard Technology, Chicago, IL (10–11 May 1999) http://www. cl. cam. ac. uk/Research/Security/tamper* (1999).

[7] MANGARD, S., OSWALD, E., AND POPP, T. *Power Analysis Attacks - Revealing the Secrets of Smartcards*, 1 ed. Springer, June 2010.

[8] NOHL, K., EVANS, D., STARBUG, AND PLÖTZ, H. Reverse-engineering a cryptographic RFID tag. In *Proceedings of the 17th USENIX Security Symposium* (July 2008), USENIX Association.

[9] RANKL, W., AND EFFING, W. *Smart Card Handbook*, 4th ed. Wiley Publishing, 2010.

[10] ROSS, R. J., Ed. *Microelectronic Failure Analysis: Desk Reference*, 6 ed. ASM International, 2011.

[11] SUGAWARA, T., SUZUKI, D., FUJII, R., TAWA, S., HORI, R., SHIOZAKI, M., AND FUJINO, T. Reversing Stealthy Dopant-Level Circuits. In *Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded Systems — CHES 2014* (Berlin, Heidelberg, Sept. 2014), Springer-Verlag New York, Inc, pp. 112–126.

[12] TARNOVSKY, C. Hacking the Smartcard Chip. In *Blackhat DC 2010* (Arlington, VA, Feb. 2010), Flylogic Engineering, LLC.

[13] THOMAS, O. Hardware Reverse Engineering Tools: New Threats and Opportunities. In *RECON* (Montreal, QC, Canada, June 2013), Texplained SARL.

[14] van Woudenberg, J., Witteman, M., and Menarini, F. Practical Optical Fault Injection on Secure Microcontrollers. *Fault Diagnosis and Tolerance in Cryptography, FDTC 2011* (2011), 91–99.