POLITECNICO DI TORINO

Master Degree in Electronic Engineering

Reverse Engineering Techniques for Hardware Security Analysis

Supervisor:

Prof. Maurizio Zamboni

Candidate:

Danilo Velludo

April 2018

Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvnient tomber entre les mains de l'ennemi

Auguste Kerckhoffs, *La cryptographie militaire*, 1883

We don't need the key, we'll break in

Rage Against the Machine, *Know Your Enemy*, 1992

Contents

List of Figures

1 Abstract

This thesis has been realized during my internship and after at Texplained SARL, a French riviera based company specialised in hardware security and IC reverse engineering.

Created in 2013 and based in Sophia Antipolis, Texplained is the reference in Integrated Circuits Security. Our company offers a broad range of services and products to educate, evaluate and protect ICs against piracy and counterfeiting. Equiped with cutting-edge machines and proprietary tools, we accompany our customers at every stage of the lifecycle of their electronic solutions: Design Review, In-depth security analysis, Backdoor Research, IP infringement investigation. We also design disruptive Secure Digital IPs. Our customers, major actors in the semiconductor area, specialize in diverse fields of applications: e-Government, Banking, Automotive, Medical, IoT, Pay-TV, Gaming, With our unique position on the market, we are able to masterize every chips and their security.

The aim of this work is giving an overview of the most recent technique used to assest the level of security of ICs by means of a reverse engineering based invasive attack. This emergent technique is based on the automation of image processing by means of a higlhy advanced custom software able to extract all interesting chip features from in-house taken SEM images.

As a case study, the method used to extract the ROM of one of the most secure microcontroller on the market is proposed and compared against an older, fully invasive technique. This secure microcontroller was never attacked before and has a EAL certification equal to 6+, the highest security level available.

The approach adopted is a full reverse engineering analysis aimed at the security circuitry around the ROM. Encrypted data have been retrieved by means of a custom software tool able to extract features from SEM images and output the whole memory content. Thanks to the decryption circuitry model, it was possible to successfully descramble the plain text code written in the ROM. Automation plays a crucial role as to effort and cost needed to perform the attack. Conventional manual tracing and features extraction would cost months of work of highly skilled engineers.

The automated approach reduces incredibly the costs and the time needed for the analysis. The output of the extraction is a full VHDL netlist description of the chip.

As it may seem clear, automation is going to play a crucial role in the hardware security field. What only some years ago was considered to be too time consuming and expensive, and so not seen as a threat, can now be performed whithin the blink of an eye and for a fraction of the original cost, thus raising the alert on actual security of devices designed without considering what kind of forces act nowadays. Though, this could desirably bring a change in design paradigms for applications in which safety is critical.

2 A kick off on Security

2.1 Some definitions about Security

The term Security stands for a wide variety of a design properties. Security is the capability of a system to protect all of its assets from an attempt of copying, damaging or making them unavailable to legitimate users [1]. Any system designer should take into account which one is the best method for defending from malevolent attacks its resources, according to their importance. An **attack** is the act of trying to retrieve or compromise a resource you don't have the right to access [1]. An attack may be aimed at compromising the **confidentiality** of a resource, meaning the possibility to copy or steal it, or its **integrity**, meaning the capability to modify its value. A set of passwords

inside a read only memory zeroed in some way would grant full access to a chip's secerets. As a matter of fact, security implies that it is not enough to prevent a secret value to be found end extracted, but must also be impossible to modify [4].

Security is, in the end, always a matter of hardware. Hardware embeds security mechanisms with much smaller performance penalties than software. Memory protection techniques rely on hardware to strenghten functions isolation. Cryptographic accelerators use hardware parallelism to speed up protocols. Moreover, hardware provides a trusted environment for computing: if the hardware is secure, then a secure system for authentication can be set up. Given that the hardware is compromised by some kind of attack, then the entire system would be insecure, since, for example, access rights checks could be bypassed. Coprocessors dedicated to security, running criptographic algorithms need to be protected at the hardware level to provide a secure computation platform. It must not be easy to access them from the outside: if an attcker can deactivate them, for example deactivating their own dedicated clock line, then it is as if they were not even there. Even if security countermeasures are adopted, if an application contains an exploitable vulnerability, malicious code can enter and compromise the system. Hardware can protect the system from software vulnerabilities being exploited by protecting an application from attacks on the software level. This is why hardware security is higly relevant and why it's hardware the preferred target of the most succesful attacks [11].

Complexity itself can in a way protect a design, making it difficult to perform an hardware attack. Modern CPUs may be a good example for this. It is difficult to inject faults in processes related to security algorithms execution or authentication by a clock glitch attack, since clock is managed internally and glitching at high speed needs expensive equipement; large data bus are difficult to microprobe and, at the cost of additional area and logic, busses can also be encrypted so that even a succesfull microprobing would not result in a full key or passwords recovery; many pipeline stages render power traces tricky to read so that power analysis related attacks are not so easy to be performed,

and so on. Brand new attacks classes like Meltdown and Spectre, though, show how the more the number of features are embedded in a processor, the more backdoors and weaknesses, even unpredicted ones, are likely to appear in the end. However, any pirate could easily tear apart a complex system and foucs its research on a particular component: in the end, the overall system's security depends on its weakest component's one.

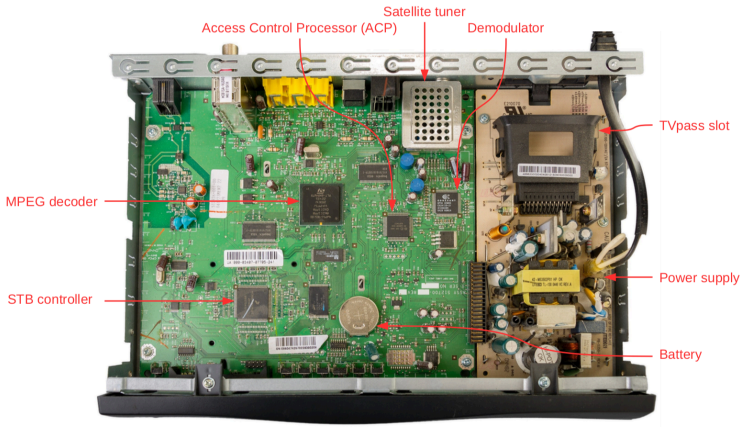


Figure 1: Opened Set Top Box. Reading power traces on the entire device is not feasible, it is advisable to look at each chip separately. Source: [7]

A system cannot be designed to resist any kind of attack. In this perspective, it becomes of particular relevance to evaluate very carefully the threat model at the design phase. In this way, it is possible to optimize in terms of silicon area the countermeasure to be taken against the most dangerous kinds of attack for that kind of application. Breaking the security of a secure system is always just a matter of time and money. Countermeasures, as a matter of fact, are designed and implemented to protect a set of secrets against a subset of attacks by rendering impractical and antieconomical the attack itself by raising its cost.

All secrets have a value, the cost of the attack must be higher than this value while countermeasure's cost must be lower. The defense solution is considered successfull when attackers can only use a strategy that takes so much time and so much money for the asset to be retrieved that they decide to move to

another target.

The notion of security can sometime clash with reliability [6]. For purpouses like testing, debugging and failure analysis, manufacturer need to add backdoors, which are undisclosed ways to access the system. Backdoors are usefull tools for all those devices that need memory backups and parameters changing over time, like most of the industrial equipment that need for periodical recalibration and firmware updates. These backdoors give almost unlimited rights to memory content access and are terribly dangerous if exploited by an attacker. It is vital to keep their existence secret, but a full chip reverse engineering analysis could find them anyway. The modern tendency of having connected devices accessible via Internet allows for large scale remote attacks. As will be mentioned in section 3, backdoors used to introduce malicious software modification can cause irredimable damages to hardware.

Adding the cost of security countermeasures in the already complex equation of choosing components at design phase can result in a very tricky evaluation task but the costs of an unsecure device for a security sensitive application are for sure unvaluable. The design of secure hardware is not an easy task for manufacturers: always new and more complex attacks are discovered, for which more and more elaborated countermeasures must be implemented at any new generation. Designers has to face all threatening attacks on one single device and simply adding up all the countermeasures would result in an uneconomical product [22]. Some smart card secure chips were recently upgraded to a newer version with SRAM based PUFs, used to create a key from the initial value of SRAM cells. 25k cells have been added to devices made up of 50k gates: this simple and not so effective countermeasure costed a 50% area overhead. Moreover, it is not easy to decide the parts of the chip that should be protected against attacks, since only taking into account for security only the cryptographic related hardware or memory blocks is typically not enough. Besides security functional requirements, designers still has to respect other additional constraints [22]

- Area: ICs manifacturing costs are somewhat proportional to the fourth

power of chip area. Customers pay for the product independently on the size, so, to make it profitable, designers must care for the countermeasures not to occupy too much additional silicon area.

- Power consumption: cryptographic engine consume a lot of power. Active security countermeasures unavoidably end up with additional power consumption. It is a must nowadays not to let the power consumption become too big, especially in the case of mobile systems. Systems relying on a contact for power supply, like SIM cards, usually have to respect precise and already high restrictions concerning power, like any device dedicated to mobile applications; contactless systems do not work if power required is higher than what the reader's electromagnetic field provides.

- Forward security: the possibility that a new attack becomes available must always be taken into consideration. Countermeasures against new attacks discovered when chips are already deployed can only be dispatched as software updates, and may not be enough or cause an unacceptable performance penalty. A ROM containing too weak keys cannot be replaced, for example.

- Reusability: since a hardware manufacturer usually does not provide the whole system, it can not implement an overall security implant, because the final applications are not known. A designer would like to design one hardware for many applications and sell it to different customers: ecxcept for some details and small adaptations, the same hardware will be provided for several uses with different requirements. An optimal trade off between functionality and costs has to be found.

- Patents and IPs: countermeasures that are already patented will be avoided because of additional licensing fees. Sometimes, particularly in low cost products, these fees could raise too much the price of the device.

All of theese are just for saying how important is, during the conception of secure hardware, being able to evaluate the security level of the final product from the very beginning. Masks costs for chips are quite high, trial and error is not an available option. Decisions need to be taken during design phase and need to be quite confident. The risk would be either overdesigning the product [22], making it too expensive, or underestimate the attackers' ability to breach into a weak device, jeopardizing its own reputation.

2.2 Some examples about Security

The most secure systems that we can take as examples are secure microcontrollers, traditionally embedded in smart cards. Applications of smart cards are always expanding thanks to the increase in storage and computational capabilities of modern ICs. [12]. Smart cards represent an embedded secure computational environment, able to keep safe sensitive informations. Since smart cards are considered the state of the art of embedded hardware security, we can take them as good example of the features a system has to integrate to be considered secure. Security expedients developped in smart card industry have always been leading the edge of hardware security for the whole silicon industry as well as all tecniques and methods for extracting informations from smart cards are common to all attackers' targets [10].

2.2.1 Brief Smart Cards history

We can read a very well detailed history about smart cards in [12], where the need and the search for higher security is highlighted as the main driver for innovation.

The first smart cards appeared in the US at the beginning of the '50s. By that time, though, cards were not so smart, considering that they were made of PVC, whose low prices allowed production in large scale. The first card payment system was put in place by the Diners Club, an exclusive credit card company that gave for the first time the possibility to its elite customers to pay for travels and entertainment without cash. The inventor, a businessman

called Frank McNamara, was so embarrassed his wife had to pay dinner at a restaurant in New York city because he had forgotten his wallet home, that he decided to solve such an issue once for good.

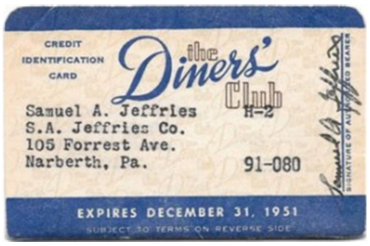


Figure 2: One of the first travel and entertainment card issued by Diner's Club, 1951. Source: moneypeach.com

In a few years, credit card circuits like Visa and MasterCard popped out, leading to the spreading of plastic cards as a payment method at first in the US and then also in Europe and the rest of the world. Nowadays hundreds of millions of cards are released every year and are accepted everywhere in the world without boundaries of currencies. The first cards had very few functions. They just needed to bring and protect from forgery the holder's name and account number, which were stamped in relief and sometimes also carried a signature. The security of the system was essentially based on the trust holders had from people accepting their payments. Quite early, the wider diffusion of cards and threats from organized crime proved that just a signature behind the card was not enough to avoid cons. It was clear that an automatic system was needed to face managing costs as well as better security functionalities to avoid frauds. Digital data were embodied in the card for the first time with a magnetic stripe on the rearside. In this way, payment data began to processed electronically. A new form of authentication needed to be introduced. The Personal Identification Number, PIN, was born, a secret numerical code associated with the owner's reference number to authenticate a legitimate payment. Magnetic stripes embedded in cards are still widely used nowadays in electronical payments. Unluckily, magnetic stripes do have a critical fault: data can easiliy be read and modified, making them a target for

tampering and thus not a suitable carrier of sensitive information. Confidential data like the owner reference number should be stored outside of the card, but this would mean having a sort of terminal able to execute offline authentication in a secure way.

The idea of an embedded system running identification opearations came to 2 german engineers, the rocket expert Helmut Grotrupp and Jurgen Dethloff, who patented the first actually *smart* card in 1968. Anyway, we must wait for 1974 and the french engineer Roland Moreno to see the first real, smart card chip as we know them nowadays. The great improvements of electronic industry at that time made it possible to integrate on a single piece of silicon of a few square millimeters all the logic and data storage needed for authentication at the suitable price. Moreno made the first demonstrative electronical payment with his card in 1976, using a machine he made on his own. The french *carte a puce* took only 8 years to wide spread all over the country, before any other countries in the world, and even go beyond the aim it was intended at first, giving birth to the first reliable telephone card.

Figure 3: Smart card by T-Mobile embedding Moreno's chip. Source: Wikipedia.com

It was clear then that, apart from being secure against tampering, smart cards provided an incredibly flexible framework for all sort of new applications. EEPROM chips started to be implemented in cards, to avoid the need for an exernal programming, for newer telephone card generation built in Germany in the 1980s. The success of telephone card was incredible: by 1990 a total of 60 milion cards were circulating in France and Germany.

As to banking applications the path of smart cards were quite more rough.

Far more complex authentication and protocols were required to ensure security over transactions. Cryptography made giant steps during the 1960s, but only in the 1980s hardware was capable to implement sophisticated algorithms at a reasonable cost and in a convenient silicon area. Smart cards were the perfect environment: they provided a high security level in a package small enough to fit everyday life's needs. It was finally possible to overcome the flaws of magnetic stripes by integrating cryptographic algorithms and secret keys in smart cards chips. Once again, the first attempts was made in France. First trials were conducted starting from 1984 and by 1992 french banking system introduced at national level the *Carte Bleue* payment card. In the meantime, in Austria, POS enabled smart cards began to be issued, making Austria the first country to have a national electronic purchase system. National electronic payment systems were established all over Europe since then, based on the European EN 1546 standard. In 1994 EMV standards were established to guarantee hardware compatibility with all major payment systems, setting the road for developements of smart card chips in the following years thus helping the spreading all over the world for this technology.

The rise of Internet payments demanded new security measures. Electronic signatures were introduced in all countries, following the related 1999 European directive.

Smart cards have proven to be the perfect fit for medical and health insurance application as well as passports and ID documents, providing the optimal balance among confidentiality, tamper resistance, reliability and durability.

During the 1990s, PayTV industry has for sure pushed the most on smart cards security, as we can see in image 4 from [10]. Huge investments made brought to a whole new level both awareness and know how about security, urged by the need to face the threat of fierce pirates. Between the end of 1989 and 1997 the popular pay-TV network Sky had to change its smard cards 11 times to face constant piracy [3]. In 1993, video signal began to be scrambled with a quite simple scheme that was broken by brute force in just a few months. In 1994, nanoinstructions were embedded in the chip of smart card Sky09 to change the behaviour when pirate cards were spotted. Useless to say, the whole

code was dumped and pirates started to update their own firmware to bring back to life disabled cards. The new version of the smart cards, issued in 1995, was made of a SoC with an ASIC dedicated to security. Once again, this time after only 3 days, a firmware bug allowed pirates to reproduce the code and sell a modified version of the original card. When kill signals were sent at 21V, pirates found ways to prevent any high voltage signal from reaching the card. Kill signals disguised as software commands were avoided by disabling in the code every command involving the card's serial number. When in 1997 the first chip with a mesh protecting the ROM, inside DirecTV P2 card, pirates started using FIB edit to access the code and exploit routines to dump the EEPROM. Code began to be written in EEPROM blocks with internally generated Vcc, internal clock monitors were studied and designed to prevent the clock from being slowed down for memory read outs [2]; *glue logic* design was introduced to make chips CPUs look like a sea of gates, were all functional blocks are melt together so that no easy target module can be spotted; crypto engines are now integrated on modern smart cards, to flatten power traces and avoid differential power analysis attacks.

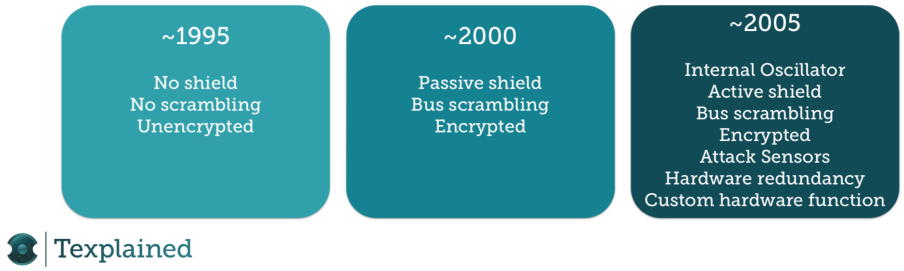


Figure 4: PayTV smart cards security features evolution. Source: Texplained

2.2.2 Countermeasure and Precautions

As we can see in figure 4, the major features of a secure device nowadays are

- *Shield* to stop a microprobing needles from reaching the inner layers

- *Bus Scrambling* to obfuscate memory content and obstacolate interconnects reconstruction
- *Memory Encryption* to protect data even when they are succesfully extracted from a memory block
- *Internal Oscillator* to prevent that an externally controlled clock may take over the entire system

but many other tecniques can be adopted against all different kinds of attacks. A huge and detailed list of countermeasures and precautions can be found in [12]. Here we will just have a look at the most important ones, just to give an overview at some tricks that have been developped and are currently used to make pirates and attackers drop the grip on a target device. Each of theese precautions has been developed in the field of smart cards, but can be used (and is actually used) to protect any kind of electronic device requiring some sort of security.

- *Technology* Features size on the chip (lines width, transistor channel length) should be near to the edge of technology. A very thin metal line is rather difficult to probe as well as very tiny polysilicon gates are tough to photograph with a good resolution. As technology scales down the interlayer is thinner and thinner, making the delayering steps harder and harder. However, the fact that at some point nodes will not get smaller will turn up into an advantage for any attackers.
- *Design Tecnique* Standard cells are used in designing digital logic core and modules. The advantage of standard cells is the possibility to use the same catalog for many different devices. The modularity and reusability of standard cell library design method, though, poses some concern about security: since the same library is shared among different devices, once an attacker has a discrete knowledge of the cells used, any further RE will be much easier. Full custom logic design makes it much more difficult to reverse engineer it but comes at a much higher cost. A solution for this

could be standard cell *Camouflage* [25]. Obfuscation can be used to trick an automated reversing tool into a wrong cell function assignment, thus making more difficult and slow the extraction of the netlist. In figure 5 a 2input NAND and NOR gates layouts are shown. As we can see, Metal 1 layout can be arranged to look identical in both gates so that it is more difficult to find their respective functions.

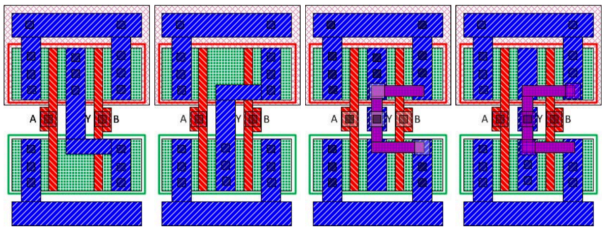


Figure 5: 2 input NAND and NOR gate, normal layout on the left, camouflaged one on the right. Source: [25]

- *Dummy Structures* Elements in the chip with no actual function at all, which are just put there to mislead an attacker. The disadvantage is extra area occupied with related costs.

- *Scrambled Bus Lines* Usually chip busses are realized in the inner layers of the chip, to make it difficult to access them from the external. Busses are usually scrambled to confuse and mislead an attacker and to hide the function of each line. Bus lines coming out of a memory block are not traced in order but are arranged randomly next to each other and distributed in several layers of interconnects. In this a way, an attacker will have a long and tedious work to do to rebuild the net of connections between addresses or functions. Chip specific scrambling increase the level of security provided. Unique serial number can be used to drive a randomizer circuit that set the chip specific bus scrambling. A variable value as input for the randomizer makes it possible to implement session specific scrambling. Theese countermeasure can make life particularly hard for an attacker trying to follow and figure out specific functions of lines in the bus. An example of scrambled and obfuscated bus can

be found in [10], where a 32 bit Flash memory output was linked to an 8 bit core. Many lines were in fact only dummy structures not linked to any register. Multiplexers were used to guide the proper signals to the Instruction Register. In this way, the attacker is forced to lose some time following useless paths without knowing which ones were actually connected to the core.

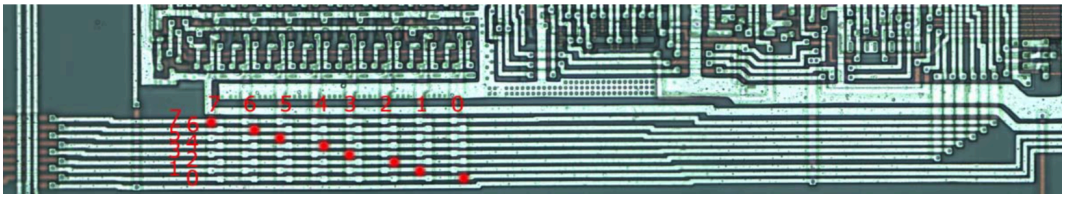


Figure 6: Motorola SC2728 secure chip, lines of the bus are not scrambled but linearly ordered. Source: [13]

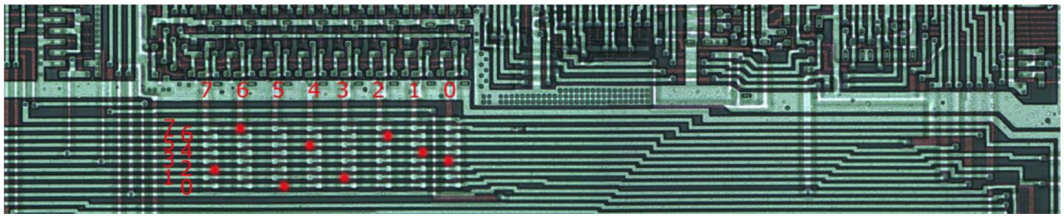


Figure 7: Motorola SC49 Smartcard MCU scrambled databus. Red spots show the proper connections of the lines to the instruction register. Swap order is: IR[7,6,5,4,3,2,1]=Bus[6,2,4,1,0,7,3,5]. Source: [13]

- *Memory Scrambling* Addresses in memory do not follow a linear arrangement but are swapped and mixed. Memory scrambling is easy to implement and does not require additional area. Software scramble makes it possible to have a swapping scheme unique for every chip and even dynamic, so that memory content is redistributed during operation. This, though, comes at the cost of more complex programming, for which all write accesses must be carefully verified and require swapping code to be

protected also at hardware level.

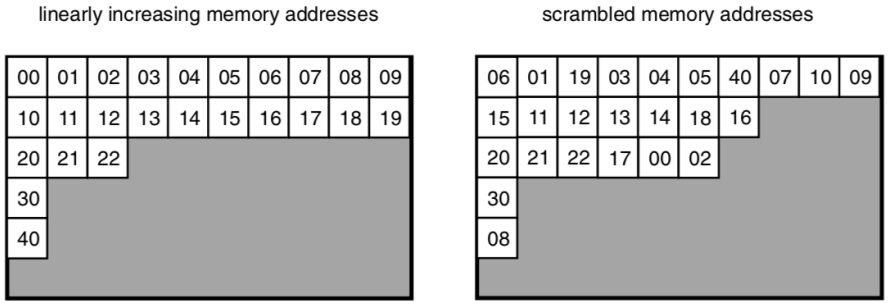


Figure 8: Linear vs Scrambled addresses mapping in memory array. This simple countermeasure would not stop a determined attacker but for sure will render the extracion more time consuming. Source: [12]

- *Memory Encryption* Data whithin the memory are encrypted. This means, though, that memory content need to be decrypted when read and encrypted when written, causing an inevitable time penalty. Encryption is made by hardware, with a dedicated circuitry implementing a function of a secret key and, in some cases, also of the address. In this way, even if bits in memory are extracted and the addressing scheme is reversed, there would still be the need to decipher encryption. Encryption circuitry is usually realized in the core, and since there exist no information about it, a full core reverese may be necessary, rising considerably the amount of time needed for a reverse engineering based attack.

- *Shields* Protection metal meshes added on top to prevent microprobing and FIB editing. Shields are meshes through which current is flowing. If a mesh line is shortcircuited, it means that a microprobing attack is happening. The simplest thing to do in this case is killing the chip or reset to zero all memory content.

  Usually sold as quite expensive IPs, they are not as effective as it may seem. Some major design and implementation flaws have been found in most of the shields on the market, reducing their strength. For example,

in many cases a shield opening only sets an alert flag in a register and software is expected to take care for the memory to be zeroed, which could be easily avoided [21].

Many shields do not cover the whole IC surface. Unprotected areas could be exploited by an attacker to get to the layers below with no need to bypass the shield. Image 9 shows a shield with a wide open area right above a RAM memory block.

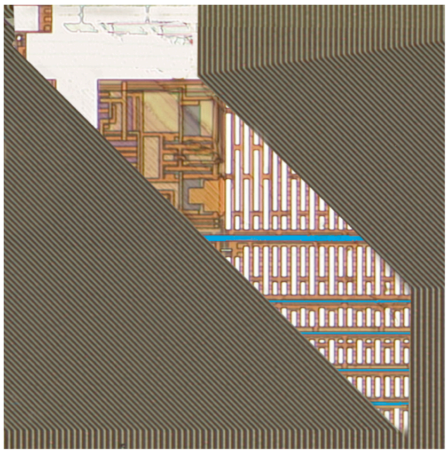


Figure 9: Shield with large uncovered area over a RAM. Nets below uncovered areas should not be important targets for attackers. Source: Texplained

Coverage is also a matter of shape. The common oval shape shown in figure 10, for example, reduces actual coverage and create areas where it is easy to pass through the shield without triggering it.

With the right tool and just a little time, any shield can be bypassed. The resolution of a focused ion beam, infact, is high enough to pass through the spacings of the mesh lines, making it possible to reach signals buried below layers of metal and create a tungsten contact. During this kind of bypass, FIB is used to reconnect the lines around an uncovered window through which the IC is attacked, as shown in figure 11.

In the end, shields do not prevent invasive attacks but elong the time needed for FIB editing (and so its cost, in some cases) thus reducing the

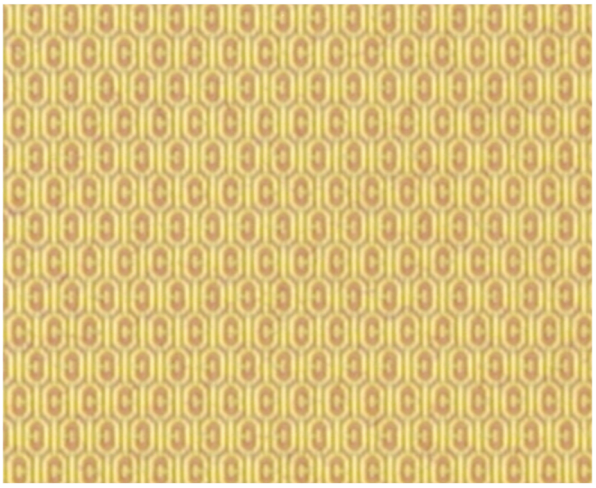


Figure 10: Oval shaped shield. The central section can be removed. Source: Texplained

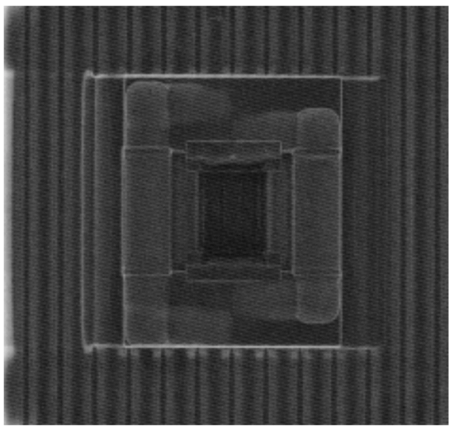


Figure 11: SEM image of a local bypass window of the width 2 mesh lines. Source: Texplained

number of attackers.

- *Voltage Monitoring* Modifying the voltage supply value over or under operating limits is one of the easiest ways of getting an IC into an unpredictable and unreliable behaviour. In this state, computations are erroneous, some instructions may be skipped: with a bit of luck, one can retrieve keys or even better, directly skip all the authentication and

security related operations. A voltage monitor prevents the chip from operating in improper voltage conditions. If the supply voltage exceeds its allowed limits, the chip is immediatly deactivated. Voltage monitors must be fast enough to detect voltage glitches, in order to protect against fault injections due to voltage glitching attacks.

- *Frequency Monitoring* Controlling the clock from the outside gives attackers the possibility to control the flow of instructions executed by the microcontroller. The chip can be forced to execute one instruction at a time, allowing to measure currents and voltages on the surface during operations. In this way, attacks based on power analysis can be put in place. A too high frequency leaves the datapath with not enough time to perform operations, introducing errors: from many wrong results of the same criptographic calculation, a fault injection attack can be set up. A too low frequency instead can induce the chip in some unpredictable behaviour, like completely jump some instruction of the code: if it is an authentication related operation to be avoided full access to the chip's secrets is granted with no effort. In order to prevent such attacks, a component that detects underfrequencies and overfrequencies should be added. Clock monitors can be set to raise an alert to the whole system when frequency is being manipulated dangerously from the outside. The system can then be reset or memory content zeroed when an irregular clock is detected. Internal oscillators and clock management units would provide a good security against these simple yet powerful attacks. However, clock monitors are often licensed as IPs for which a manifacturer has to pay for.

- *Temperature Monitoring* The capability of retaining logical values for a certain amount of time without power supply is called data remeanacence. The time during which data are maintained in RAM memories increases when the temperature decreases. Very low temperatures allow to extract RAM values, giving a complete acces to data used during computation. Temperature sensors integretd on the chip would to de-

tect abnormal environmental conditions and cause the erase of all the memory content.

- *Irreversible Switch among Test and User Modes* Test mode used for testing during the fabrication process must not be accessed by the user. In test mode, EEPROM and Flash full content can be accessed and read and all configuration settings are accessed. After testing phase at the end of manifacturing process, it must be impossible to switch from user mode to test one. This switch can be realized by means of a fuse. Fuses are essentially transistors, often organized as a memory blocks, which have been burnt by a high voltage so as to be stuck at a high logic level when programmed. Fuses are relatively large structures that can be easily spotted during a reverse engineering analysis and bypassed with an FIB edit, in order to set the device in test mode and get its whole memory dumped. Alternatively, a fuse can be made conductive again by means of UV rays. Since fuses essentially store a value programmed once for all, they can also be used to store keys, passwords and indentifiers, in addition to information on data acces rights, which make them particularly interesting as targets of attacks and analysis. It can be made particularly difficult to find the location and the value of a set of fuses with common and simple tecniques like optical or microscope inspection by placing fuses array inside main memory blocks with shared control lines. Tecniques from Failure Analysis should be implemented but all require equipement and expertise. In this case, using a UV attack to disable a fuse based protection would mean damaging the main memory array. Even better security is achiedved by using some memory cells as if they were fused, since it is very difficult to understand which ones are fuses inside the main memory array. A full reverse engineering of the memory would be necessary to find out how to deal with the fuses and to strategize an attack, which is a not exactly whthin the reach of everyone. One of the drawbacks of fuse based security can be noticed in figure 12, where residuals left after fuse programming are shown. An expert's

eye would easily spot the position and the programming state of such fuse, compromising the security mechanism it leverages. Moreover, fuse programming is not always effective and sometimes a blown fuse cannot be read as a '1', thus lowering the overall process yield.

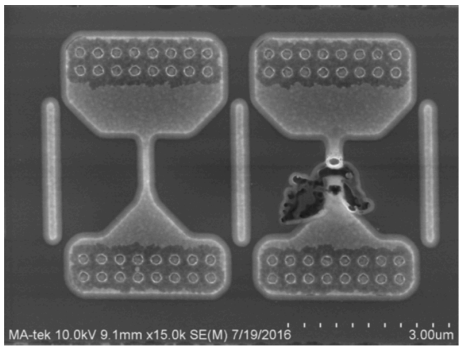


Figure 12: A programmed poly eFuse.This fuse is programmed by electrically blowing a strip of metal or poly with a high current using I/O voltage. High density current leaves its marks on silicon. Source: [14]

The answer for this can be antifuse technology. Antifuses are programmed by means of a high voltage applied to the gate that induce a breakdown avalanche that short circuits gate and source. This mechanism allow the programming operation to be performed more than once, almost 20 times, resulting in an higher yield for the process, since if the fuse can't be read from the very first time, it can be programmed again and again. As to security, an antifuse offer a higher level of protection, since there is no visible difference among a programmed and an unprogrammed one [14].

- *NVMemory Implementation* Memory blocks implementation as well plays a delicate role in the balance among cost, performance and security. Non volatile memories such as ROM and Flash memories contains code and other sensitive data like passwords and keys. Full memory content extraction is a serious threat for any embedded device. Unlike Flash, Mask ROM does not have any interface for external access to internal data. An attacker should read optically the data from the memory itself, which

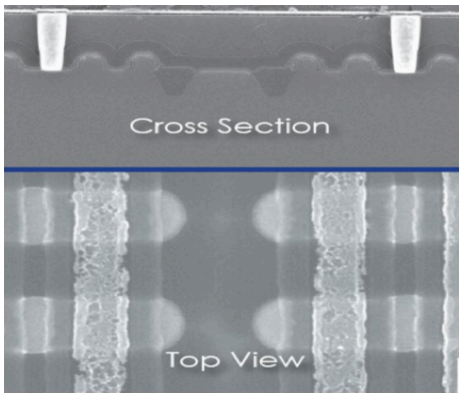
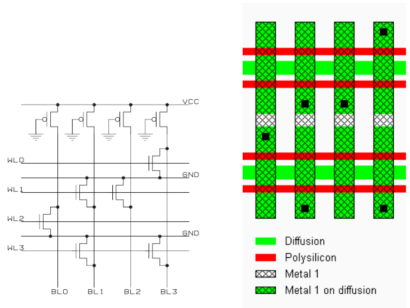


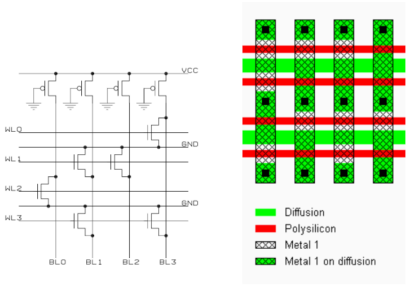
Figure 13: A programmed polysilicon antifuse. Source: [14]

costs time and skills and need equipement. According to either NOR or NAND Mask ROM configuration, data are stored in a different architecture. In NOR ROM information can be encoded either on the active layer, using threshold raising dedicated implants, or on the contacts layer. It is also possible to store ROM information on more than one metal layer connections, even this last solution is rarely implemented. Reading a ROM memory requires always some deprocessing. In NOR ROM with active layer programming, figure 14b, information encoding relies on the presence or less of transistors between GND and bitlines. It is not difficult to extract this kind of memory by optical analysis since transistors are easily identifyable on the polysilicon layer. In NOR ROM with contact layer programming, figure 14a, transistors are placed on every wordline but only the programmed ones are connected to GND through contacts. For modern technologies, this kind of memory seems to offer a slightly better level of protection since a more careful deproccesing is needed. The choice among theese two kinds of memory, from a designer's perspective, depends on a trade off among area/ performance and time to market [8]: active programming allows an area spare of about 15% with respect to contacts programming. However, contacts programming allows to process many wafer untill the step of VIAs definition, which is usually done at the end of the entire process, and burn the memory

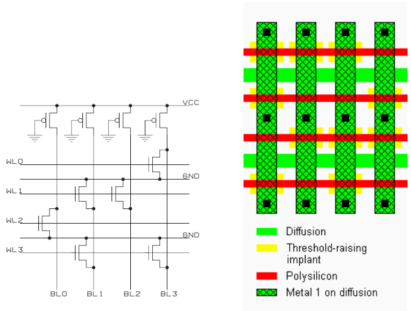
as soon as its content is defined. In NOR with raising threshold implants, figure 14c, data are stored by modifying the doping level of the programmed transistor. This means adding dedicated step in the process for threshold voltage adjustment but results in a more secure memory, since it is more difficult to read these features after deprocessing. More sophisticated tecniques are needed to extract this kind of ROM. A drawback of implant based ROM encoding, which is currently debated, is the impossibility to spot eventual backdoors inserted during production by a malicious manifacturer.



(a) NOR ROM encoded on contacts. Source: [4]



(b) NOR ROM encoded on active areas. Source: [4]



(c) NOR ROM encoded by threshold raising. Source: [4]

In NAND ROM with contacts layer programming bits are programmed by short circuiting Source and Drain through Metal 1 connections. Since no GND connection is made inside the cell, this architecture allows a more dense structure but this kind of memory can easily be read optically. NAND ROM programmed with threshold voltage raising implants cost

an additional step in the process but allows for both a more dense and secure design.

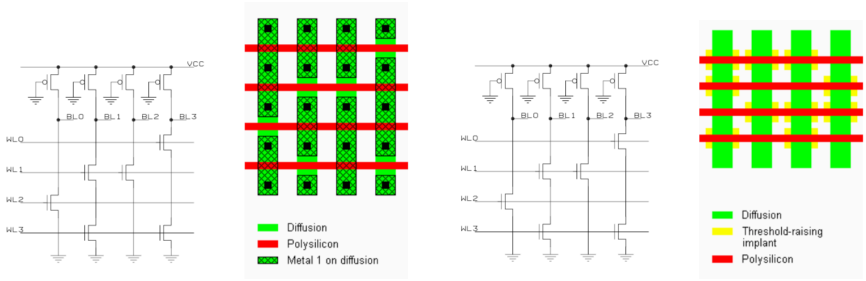


Figure 15: Different implementation layouts of a NAND ROM: implant based ones are the toughest to be extracted. Source: [4]

All Flash memories offer a good security level against invasive attacks. The charge detained in the cell is so little that an atomic force microscope is needed to probe it. Other ways can be exploited to extract theese memories, that rely on firmware loopholes and backdoors of the code.

- *Destruction of Test Circuitry* Special test circuitry is integrated on th chip to get access to busses and important lines like Clock and Vcc, to verify their state after the chips come out from production. The test logic is accessible through dedicated test pads on top metal layer. Normally, on common microprocessors, the test circuitry remains on the silicon after test phase. In secure microprocessors, instead, some fuses are blown on the lines to close them and avoid access to test module, as in figure 16. However, since attackers can reconnect the lines by FIB editing and use test module to read the entire memory, test circuitry is often phisically destroyed by the manufacturer. A chip's test interface is placed onto the area of the following chip on the wafer, in such a way that it is eliminated when the wafer is cut into separate dies.

- *PUFs* Physically Unclonable Functions are unique properties of a device due to manifacturing processes' intrinsic variations, caused by random

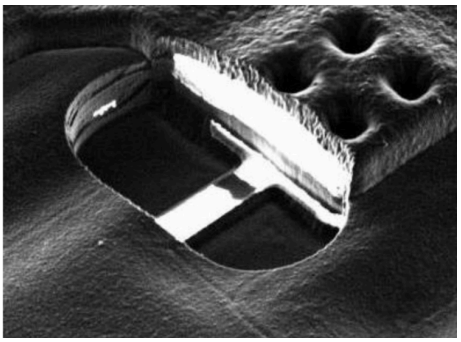


Figure 16: Blown fuse cuts access line to a test pad. Source: [21]

and unpredictable factors. They can be used as a sort of ID to distinguish a device from an identical one. PUFs are popular in secure microprocessors since they don't require any additional circuitry and can be realized in memories already present on the die. The most used PUFs are based on SRAM cells, whose settling state implements a unique challenge/response pair, issued either permanently or only at startup. Theese PUFs are particularly robust to temperature variations and extremely compact. There exist the possibilty to clone SRAM based PUFs [27], but however it requires an FIB circuit edit, meaning expensive equipment and a quite large dose of expertise are needed.

Many of theese tecniques can be put forth at the same time and combined to try to make attackers waste as much time as possible, in the hope that thay will surrender. It has to be said, though, that a determined and well equipped attacker will never surrender and sooner or later will find a way to work around any protection scheme.

3 Attack Scenarios

Before trying to protect a device against violations of some kind it is ncessary to know why and who could be interested in breaking the system. Typically, attackers have two basic motivations: the first is simply money, while the second is gaining a name in the hacker scene. In any case, if and when attacks

details become public, the reputation of the system's manifacturer is to some extent damaged. Attacks may come from people known as hardware pirates, higly skilled individuals able to break the security measures of smart cards and chipsets to give birth to illegal markets parallel to the legal ones. Most of the efforts of pirates in the years have been focused on the PayTV market, which has been the first industry to suffer from piracy [10]. Counterfeited smart cards can give access to the same contents as a genuine subscriber at a much lower price, with huge monetary damages to PayTV companies. Nowadays things have evolved and PayTV is no longer the preferred target for hardware piracy, which has moved to more profitable and less dangerous markets.

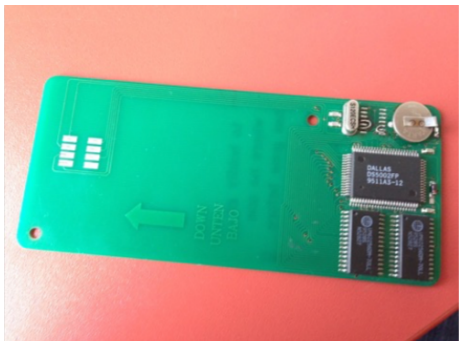


Figure 17: The battery card, successor of the Dallas card, 1995. Source: Texplained

Video game industry knows something of how dangerous piracy is. Popular video game console Dreamcast, made by the japanese company SEGA was pulled out of the market in 2000 because a German hacker group discovered a backdoor inside the Dreamcasts boot ROM [9]. The failure of security mechanisms resulted in a loss in the sales of legal videogames titles and costed the trust developers had in the company so much that right now, SEGA only develops video game and is no longer producing any hardware. Nowadays, in the video game field, a common issue is the hacking of the authentication mechanisms that pair the console with controllers and other accessories. In this way, one could produce and sell big volumes of compatible devices at a fraction of the price of the branded one. The funny thing is that this kind of

hacking is legal, as long as it does not infringe any patents, since it is meant to provide customers with a competitive alternative. The same holds for printer cartridges: the market is full of compatible products with lower prices than the branded ones. These products are compatible because someone, somehow, managed to extract and reproduce the authentication procedure of the original piece. The security of hardware involves the entire supply chain of manifacturers without fabrication plants. Manifactures often design their chips but rely the production on fabrication plants oversea, where prices are lower. It is not always possible for them to make sure that their design are not being analised and copied or even worse, that some backdoors have been added maliciously during production [11]. The reduced number of actual chip makers in the world is itself a concern for security: once attackers have figure out production tricks used by one manifacturer, they can exploit their knowledge on a huge numbers of device in the market, even belonging to different designs: as mentioned in [13], since common flash layout is the same for all manifacturers that get their chip made by TSMC, understanding flash behaviour of one means knowing the same of many the others.

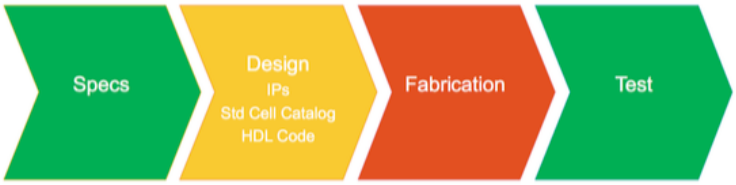


Figure 18: IC supply chain. Design houses rely on other companies for most of their work, introducing semitrusted and untrusted phases which make the entire design less trustworthy. The risk is that some vulnerabilities are introduced or some secrets stolen. Source: [11]

Attacks on the supply chain can happen during untrusted phases. Designers are assumed to be trustworthy by definition, but not all parts of the system are conceived insisde the company itself. Third party IPs, for example, are used in such a way that designers cannot verify their contents, leading to a lower level of trust for the design phase. Fabrication processes cannot be trusted

due to a lack of verifiability of offshore foundries process. IPs may be stolen and the device may be cloned during manufacturning so that there are no means for the designer to acknowledge, neither during test phase. IPs can be protected thanks to watermarking, which provide a proof of authorship over a proprietary design and allow to safely distribute and track IPs. Test phase, finally, is trusted in its entirety since is made at home. The test phase makes sure that each IC passes a set of checks for manufacturing faults. However, since it is based on design requirements, the test phase may not be able to detect hardware Trojans [11].

Attacks may also be designed within the reach of a scientific research, causing however a sort of image damage to the target's manifacturer. A university security research group will be interested in publishing the results as soon as possible and they would be available for anybody. Competitors may represent as well a serious threat to a manifacturer. It can be very profitable for a company to copy algorithms, IPs and other secrets from a competitor's device, in order to try to improve their design and build a better or a far less expensive product. For sure, huge costs related to long a delicate R&D would be utterly lowered if a fast an expensive way of unveiling a design is found. This kind of thing just keeps on happening, industrial espionage and trade secrets theft has ever occourred and certainly are not going to stop. In a way, if we want to see the good side of this phoenomenon, it is a way to spur further research and investments in more secure and reliable devices and processes. On the other side, the attacked manifacturer could be interested in reverse engineering to find out if some of his patents and IPs have been infringed by competitors to face them through legal means and save its own position on the market.

Attack scenarios can be classified according to [4] into 4 main and typical categories

- *Cloning* One of the most common cases, put forth by large companies as well as individuals who want to sell a device without big investments in design and development. It may require an in depth reverse engineering analysis. Consumer electronics market is flooded with low cost imitation

of well known brands' products, it is easy to imagine how this can happen. Just to cite an example from a completely different area where security is a great concern, it is sufficient to think about cloned credit cards black market. POS memory dumping malwares are the most used way to retrieve credit cards authentication data [17], since a point of sale is the most involved device in credit card payments. These malicioous software exploit the little period of time during which data from cards are stored in plain text in POS's RAM to get all authentication and account related data. In figure 19 we can see how the number of known POS malware is on the rise during recent years, affecting several of the main retail chains. The most dangerous malwares have the capabilty of sending stolen data using networks and even mechanisms to destroy themselves before being discovered.

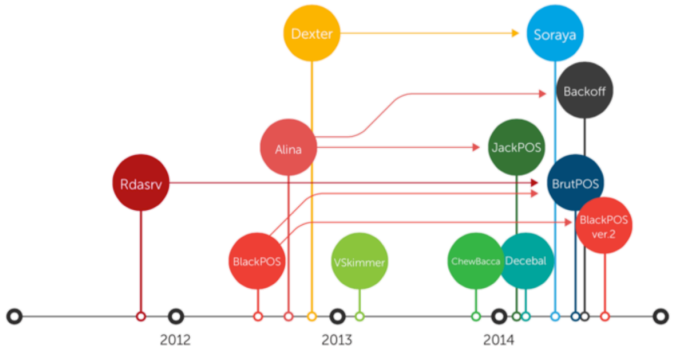


Figure 19: Rising number of POS malware is highly indicative of the spreading of credit card frauds. Source: [17]

Once the data are sent back to attacker's server they can be sold for offline carding. The buyer can easily convert the stolen data into a counterfeited card either by means of specialized illegal services or by buying for a really low price all the needed hardware. Offline carding may sound dangerous but is the kind of fraud with the highest rate of success since noone looks suspiciously a credit card payment in stores.

- *Overbuilding* Manifacturing costs raising at any new generation as fore-

seen by Moore's Law coupled with decreased production costs in the East led the semiconductor industry into a state in which the bulk of companies only design netlist while only a few manifacturer's actually produce chips. A manifacturer could build more devices than what is thought to sell the extra devices on the market or to someone interested in getting the design. Fingerprinting techniques have been developped to limit overbuilding. A hardware signature, unique for each chip, is embedded in the IC so as to easily spot illegitimate copies of the same device.TThis ca be achieved by adding some constraints to each chip's specifications so that each implementation becomes unique, thus labelling every copy.

- *Theft of Service* When en electronic device like a smart card is used to authenticate genuine subscribers of a service of any kind, like PayTV for example, attackers are interested in simulating the same kind of authentication mechanism to access that service, with no income for the provider. Just to put it with some numbers, a recent survey has discovered that in Sweden, the 29% of population maked use of illegal films and TV services, 25 % percent of UK's internet users illegally downloads movies and TV programs while in Spain only 36% of all contents in the whole country during 2015 was enjoyed legally. [18]. At first, during the late Nineties, pirates were interested in building pirated cards with full subscription plans and neverending expiration dates. Pirate smart cards and modified decoders were available everywhere, online or in retail shops. By the end of 1999, italian market was overwhelmed by cheap counterfeit Canal Plus smart cards. This phoenomeno has grown so bug that during 2002 Canal Plus Group discovered issued a lawssuit against its competitor, NDS Group, a subsidiary of Sky Group, with the accuse of being the mind behind this giant signal hacking [20].

  Broadcasts are secured against illegitimate receivers by means of a set of secret keys. A signal which is sent in the air from a satellite can be received by anyone with a dish antenna, so it must be encrypted to make it visible only for those who own a regular subscription. Since the signal

is scrambled, a key is needed to see it in plain text. The session key, or Control Word, linked to the content the user is allowed to access, allows the streamed content to be descrambled so that the viewer can see the services he paid for, for the period of time of his subscription. This key his changed regularly, on average every 10 seconds, to make it harder to brute force and strenghten the audio/video ancryption. The session key is broadcasted as well in an encrypted manner and is decrypted by means of the combination of a service key, sent encrypted in the stream, and the smart card key, written a memory whithin the smart card. Useless to say, the smart card key must absolutely be protected against tampering.

The greatest weakness of DVB CA system is relying on a permanent key inside the smart card. If an attacker manages to access the key of just a single legitimate smart card, he can produce counterfeit smart cards with the same key, since there is no feedback mechanism to control the cards apart from subscription payments and so there are no means to find out how many copies of the same cards are working at the same time. Return channels and efficient keys update was the major design update that helped current generation of conditional access systems to eliminate signal piracy. However, piracy has not ceased to exist but has rather evolved to something different. Now, thanks to the Internet, pirates can share directly the content itself rather then the keys that protect it. As broadband speeds continued to evolve, pirates moved to websites that stream content in return for banner adsvertisments or subscription/pay-per- view payments, just as the same as legal services.

Hardware prices dropping down while performance increases, broadband services getting better and better [18] make it easier to set up a pirate streming site. Open source technologies too make life easier to pirates. Open source Set Top Boxes prices generally vary from US$30 for basic models to US$150 for advanced 4K Android devices. Since they are mainly based on open source software, these STB can be sold and modified much faster than legitimate products, which must respect more

restrictive standards of stability and quality. The most famous example for this kind of products is the Linux based Dreambox STB, produced by german company Dream Multimedia. Its upgradable firmware can host unofficial third-party conditional access software modules (CAMs or emulators) that emulate the CA systems developed by NDS, Irdeto, Conax, Nagravision, Viaccess and other proprietary vendors [19]. This gave the possibility to pirates to develop a new business model, based on selling dedicated firmware upgrade for Dreambox STB to allow the device to receive via streaming a pirate PayTV subscription on a much lower price. This new type of piracy, called IPTV piracy, is currently on the verge.

- *Denial of Service* A DOS attack makes an authorized access to a system resource impossible or delayed in time. As to embedded hardware, a great source of danger can come from over the air firmware updates. Recently in fact, a HP researcher has demostrated that the so called Permanent Denial of Service attacks can happen remotely [15]. A PDOS attack is a DOS attack that permanently compromise the target, making it unusable. Usually, after a DOS attack, normal operating conditions can be restored by resetting the system or waiting for the stop of overwhelming traffic. In PDOS instead it's not possible to recover previous working conditions. Compromised hardware needs to be substitued, which is way more costly than simply releasing a new update to solve the issues and cause a bigger damage to a company's image with respect to its customers. An FPGA can be permanently damaged by simply uploading a bad configuration file [4]. Moreover, a PDOS attack cost way much less than a Distributed DOS: it does not imply the rental of a bot network to obstruct the service, after the malicious firmware has been downloaded the DOS condition will continue indefinitely. The problem with embedded hardware is that it does not get patched or reviewed often and can contain application-level vulnerabilities, such as flaws in the remote management interface that leave the door open for an attacker [16]. A

competitor able to reverse engineer the update protocol and figure out the update signature could launch a malevolent release to ruin the company's image by damaging its products. A pirate could use this channel to insert backdoors in any device, gaining total control over it. The wide spreading of embedded devices with automatic update capabilities over the internet could make this kind of attack a serious threat, though it is much more profitable for a criminal organization to exploit a network of connected devices with poor access security than destroy it. For sure, being aware of the high risks that lie in flaws of update mechanisms help in designing always more reliable embedded hardware.

3.1 Types of attackers

The most famous hacker in the world was used to introduce himself into big companies databases by means of his phone. He just leveraged onto the right person, pretending to be either an high level executive or an external consultant to get the credentials to access sensitive information. Surprisingly, he never used the data he acquired to do any harm to nobody, causing no money loss at all to anyone. A pioneer in network based attacks, he was even able to track the FBI phone calls, capability that let him avoid arrest for years. When he finally surrendered, tired of being chased, he was incarcerated with no charges: threre were no legal means to classify his crimes. Kevin Mitnick is considered the inventor of social engineering and is for sure the first hacker to gain fame and media hype ever, because of all the controversy his case went through. His story is emblematic, though: how is it possible to classify attackers? Moreover, where is the boarder between legal and illegal?

Figure 20 shows a classification of attackers [12]. All attackers can be dangerous to a secure system, but since they have different properties and capabilities, they represent diferent threats to be coped with. Hackers, for example, have usually good ideas, quite good knowledge and can count on a tight group, but have limited economical resources, for sure much less than a criminal organization, which in turns may not know the system and have the

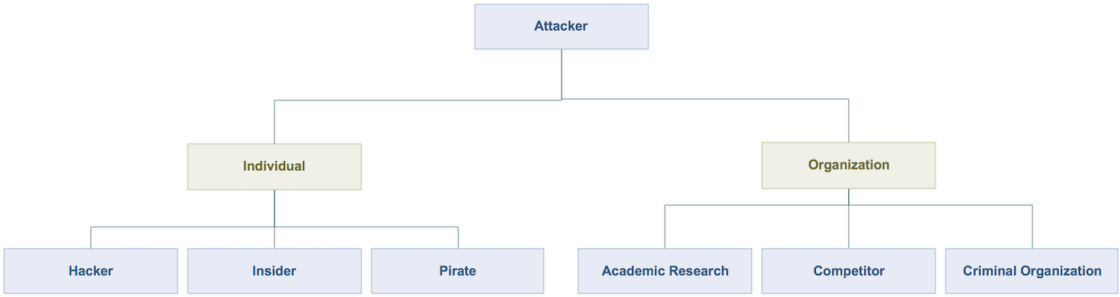


Figure 20: A chart of different attacker's types. Source: [12]

expertise to make clever use of equipment.

In [4] we can find a classification proposed by IBM which summarizes the previous one into 3 main groups:

- Class 1: clever outsiders
  Skilled people with insufficient knowledge of the system and low budget equipment. They will try to exploit an already known security flaw rather than discovering a new one. In [1] we can find a subgroup of this class defined as **Remote Attackers**, which are distributors of malicious software such as malware or viruses, with no direct physical access to the devices they are attacking. The uprising diffusion of devices able to run browser-based content makes it easier to spread malicious code: the IoT era to come will probably see the concern about security grow as the web can be used as effective attack vector. Hackers go under this class as well. It is important to clarify the difference between a hacker and a pirate. Though theese two categories may be putting forth similar attacks, we can't absolutely say the same on their purpouses: a pirate is a sort of burglar, whose final scope is making as much money as possible out of illegal exploitation of hardware; a hacker is an enthusiast who is just interested in knowing how things work and how they can be modeled into something new. There is no malicious intention in overcoming security measures, it's just the pleasure of pursuing curiosity.

- Class 2: knowledgeable insiders

Specialised technicians that know the system quite well and can access it with quite sophisticated instruments; moreover, they could know about design flaws and security weaknesses of the device. Employees that use their privileged access to steal a company's secrets can cause a great damage and represent a difficult kind of attack from which to protect: they are fully trusted until it is too late. Legal precautions like mandatory NDAs signing and strict policies on sensitive data access are taken as precautions to prevent these attacks. Theft of trade secrets usually ends up in justice courts with huge money being spent in legal actions and settlements but beside the arrangements that can be made among contendants, the damages to years of investments and efforts in R&D are difficult to estimate.

- Class 3: funded organisations
  Team of experts with great economical resources. Capable of deep analysis and design of complex attacks with the most advanced equipment. Usually thier aim is defining a new class of attack that can be applyed to a large number of devices with a much lower cost. This kind of attackers are the most dangerous as they can access all the features of a device even without any previous knowledge of its design. Here we can find universities and research groups, competitors and criminal organisations. Universities may not have special knowledge of secure devices, but have a wide general knowledge and highly motivated people with an experimental mindset, which can bring to big results even with little expenses [12]. Far more dangerous are instead competitors, who they have a deep technical knowledge and very sophisticated. Criminal organisations may not be formed by experts in the first place, but have enough resources to get in touch with people capable of conducting attacks and find equipment.

IBM also definede 6 protection levels [4] that a device can provide, from no protection at all to the (in principle) unbreakable system.

- Level ZERO: No security precautions are taken, any component is easily accessible. In figure 21 we can see a controller from Micromint Inc., where each component can be identified thus allowing the board to be cloned [4].

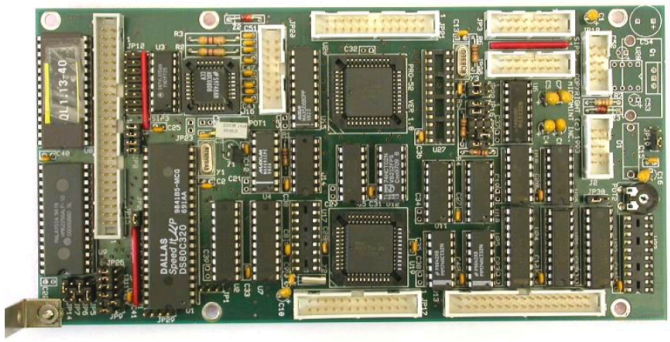


Figure 21: Embedded systems made up by separate components can be easily reproduced if the names of all components are available and traces can be easily followed. Source: [5]

- Level LOW: Some light security countermeasures are taken but an attack can be performed with less than 1000 dollars equipment.

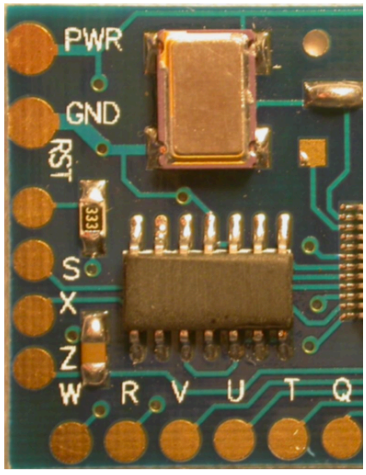


Figure 22: ASIC carrying out some I/O function, not so difficult to understand by analysing its signals. Source: [5]

- Level MODL: The system is protected against the cheapest and simplest attacks but still can be victim of attacks such as power analysis, which can be performed with around 3000 dollar equipement.

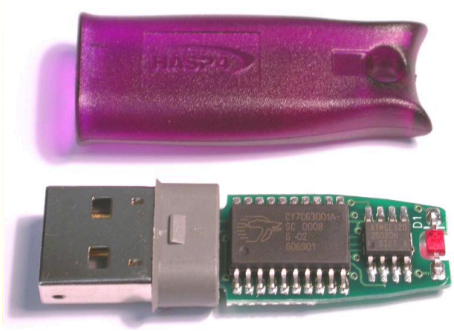


Figure 23: USB dongle with external data EEPROM, easy to access. Source: [5]

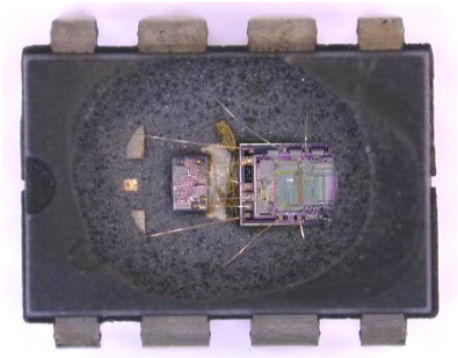


Figure 24: Data EEPROM and microcontroller packaged together, need for some deprocessing to access it. Source: [5]

- Level MOD: Attack require many skills, time and up to 30000 dollar equipement. This is the case of UV fault injection resistant devices.
- Level MODH: Quite well designed device under the security perspective. Attacks can cost more than 100000 dollars and must be put forth by a team of experts. Complex ASICs implement this level of security.
- Level HIGH: There exist no known attack able to defeat the security of this device so a new one must ne researched by a strong team of experts.

Figure 25: Old smart card, embeds some security features and needs deprocessing. Source: [5]

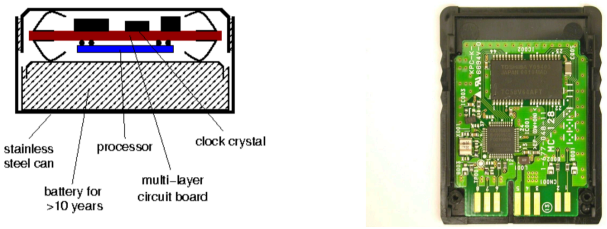


Figure 26: Secure device with internal power supply and complex PCB board. Source: [5]

> Research takes much time and can cost up to millions dollars, only a few organization are able to bear the risk of an quite uncertain result.

There exist an international standard, ISO/IEC 15408 to certificate computer security called Common Criteria for Information Technology Evaluation, abbreviated as CC. CC are the technical basis for an international agreement, the Common Criteria Recognition Arrangement, whose aims are ensuring, as stated on CC website, coherent standards for security profiles evaluation. Members of CCRA are national institutes have the authority to approve compliance with ISO/IEC 17025 General requirements for the competence of testing and calibration for licensed testing laboratories. Licensed laboratories evaluate products' security features and verify that manifacturer's security claims are effectively implemented. In France, for instance, the national CCRA member agency is the Agence Nationale de la Scurit des Systmes d'Information -

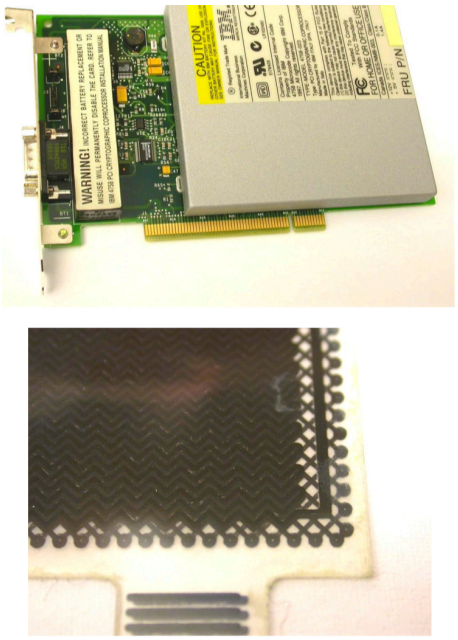


Figure 27: IBM 4758 PCI Cryptographic Processor, the first device to receive the highest level of security certification by US government, FIPS-140 level 4 [2]. Source: [5]

ANSSI, while the italian one is called OCSI - Organismo di Certificazione della Sicurezza Informatica.

A CC evaluation ends with assigning a numerical grade to the device under test, called Evaluation Assurance Level. A EAL level is reached when the device meets some particular requirements involving documentation, analysis, functional and penetration test. EAL levels are 7 and cover any IT systems, be it an OS or a microcontroller. There exist also certifications which are specific to hardware whithin the EAL, as well. The higher the EAL number, the higher the cost and the time needed for the design to get that certification. Security level associated to any of the EAL is based on an evaluation of the documentation and design details provided by manifacturer. Higher levels require a more extensive documentation analysis, indipendent testing, both functional and penetration, and proof of anti-tampering. However, tests and analysis required to the designer are conducted in laboratories chosen and

engaged by the designer himself, letting some kind of possible breaches through this system of certification. Full description of all the seven assurance levels is given in the CC document *Part 3: Security assurance components*.

EAL certification takes into account also attackers potential. CC gives the following metric to calculate the attack potential an attacker need to have to be able to perform a certian attack [22]. In particular, two parts of the attack are considered:

- Identification: efforts needed to find the weakness and design the attack
- Exploitation: efforts needed to perform the same attack on a second sample, using results of Identification part.

Factors that need to be summed up to state the vulnerability of a target against an attack are, therefore:

- Time: how many hours, weeks or months are needed for designing and performing the attack even more.
- Expertise: how much of general knowledge and know-how does the attacker need.
- Knowledge of target: what kind of information about the target the attacker need, i.e information publicly disclosed and available in datasheet or reserved information available only upon signing an NDA.
- Access to target: availability of target samples,in terms of cost, delivery time and number of samples needed for the attack.
- Equipment: machines and tools needed for the attack, in terms of costs and availability.

The higher are the parameters, the higher are the efforts to break the security of the target [22]. It is quite obvious that attacks that can be made through unexpensive tecniques and in a little time are the most lethal weapons against which one must protect.

3.2 Types of Attacks

There are many ways of grouping and defining types of attacks on a secure device. Moreover, since security evolves together with technology, always new ways of categorizing attacks are proposed as new attacks and methods are discovered and disclosed.

- Attacks on the physical level: require equipment since it is necessary to get physical access to hardware. Such attacks can be either:
  - Static: The target is turned off and is not operating. Static attacks impose no timing restrictions to attackers.
  - Dynamic: The device is on and operating, equipement must be fast enough to acquire and evaluatw data, meaning possibly newer and more expensive.
- Attacks on the logical level: this category includes crypto analysis, attacks that exploit known faults in the system and Trojans in the executable code. As to analysis of cryptographic protocols, we can distinguish:
  - Passive: attacker analyzes the encription circuitry and cryptographic protocol without modifying them.
  - Active: attacker manipulates the data transmitted or the microcontroller.

.

Here, we will mainly deal with attacks on the physical level.

3.2.1 Attack Methods

There are three main different kind of attacks [5], based on the depth of deprocessing required. The more deprocessing an attack requires, the more time and resources it will cost, in terms of laboratory facilities and expertise. However,

this does not necessarily define at the same way the seriousness of the threat posed: an unexpensive and fast attack, maybe exploiting a design flaw is of course terribly more dangerous than a long and expensive attack.

- *Non Invasive Attacks* These attacks deal with external interface signals of a device. Such things as manipulating the clock from the exteranl to trick the device into erroneous computation or analizing power traces during execution of cryptographic algorithms to retrieve the key can be done without the need of particular tools and any previous sample preparation. Package is not affected during this kind of attacks and the chip inside is not damaged. No chemical or mechanical deprocessing is needed, no damage of any kind is caused to the chip. For this reason, theese attacks are the cheapest and the fastest ones to be performed. The fact that they can be performed in a little time and with (relatively) little effort makes them quite threatening. Main Non Invasive Attacks deal with

  - Timing Attacks
  - Brute Force Attacks
  - Power Analysis
  - Glitch Attacks
  - Data Remanance

  Non invasive attacks are an attack vector to be taken into consideration in applications where tamper evidence must be provided, like banking and digital signatures, where no longer valid keys can easily be changed [21].

- *Semi Invasive Attacks* Everything that lies in between Invasive and Non Invasive can be considered as Semi Invasive. Chip is depackaged but not deprocessed: IC remains intact and fully functional. The surface is accessed but not violated for probing or FIB editing. Theese kind of attacks are penetrative and more likely to succed like Invasive ones

but are far less expensive and repeatable like Non Invasive. Technology scaling makes Invasive ways harder and harder to follow, since always more sophisticated facilities and equipement are required: Semi-Invasive seems to be a good trade-off in this perspective. Are considered Semi-Invasive theese kind of attacks:

  - UV Attacks
  - Backside Imaging Tecniques
  - Fault Injection Attacks

- *Invasive Attacks* Theese attackes are performed directly on the device components. During the attack, the device is opened and progressively deprocessed. Package is removed and internal signal under layers and of metal are accessed, either to extract a ROM memory content or to probe a data bus. This kind of attacks may be performed only by a well equipped and trained team of experts, require time and last but not least, money. Usually, aninvasive attack is the starting point for a more convenient attack: once the weakness is found, the attacker can strategize a non invasive attack or other simpler ways to overcome security countermeasures. Invasive attacks do not need a particualr initial knowledge and usually work quite in the same way with a great number of similar devices. Theese kind of attacks was not used to be considered as a real threat due to the huge time and investment they need. Industry simply knew of the endless capabilities that a complete reverse engineering can give to an attacker but since very few people had the means and the competences to afford it, this threat was kind of ignored. However, costs can be lowered renting services or buying second hand equipment. Are considered Invasive attacks we can list:

  - FIB Editing
  - Microprobing

4 Invasive Attacks and Reverse Engineering

Invasive attacks require direct access to the inner part of the device. Whether the target is a secure microcontroller of a smart card or a SoC from a video game cartridge, it must be pulled out of its package for the inner die to be analysed. Passivation layer must be removed, as well as eventual top metal layer meshes or shields must be circumvented to go through all interconnects layers untill silicon bulk, in a process called *delayering*. It is evident that only a well equipped and quite expert attacker can succesfuly carry out the first, laboratory based part of the attack. If facilities and expertise to safely stock and employ acids are not whithin the reach of everyone, chemical mechanical polishing machines are by far less accessible, and we are not even considering FIB machines. A solution for this though, would be renting facilities or outsourcing this service to a specialised laboratory. Because of mandatory delayering process, invasive attacks becomes more and more expensive and complex as technology evolves.

On the security analysis side, invasive analysis techniques are particularly interesting because they are the only kind of attacks with which it is possible to virtually overcome all known hardware and software countermeasures [24] with a 100% success rate. Reverse engineering in itself cannot be considered as an attack: it's a practice to gather knowledge about all features and properties of a design, which can then be used as a starting point for finding the most effective attack vectors, making it the most powerful tool for security evaluations. Only a few years ago, let's say ten or even less, it was practically impossible to reverse engineer even quite old technology nodes of the order of thousands of nanometers, due to a lack of automated tools for feature extraction. Netlist tracing and annotation were done manually by stitching together images and drawing schematics with pen and paper.

In the following sections, an overview of invasive reverse engineering workflow will be given and it will be shown how analysis results can be used to evaluate the risks and attacks the ROM memory block whithin the IC is exposed to.

5 Reverse Engineering

5.1 A kick off on Reverse Engineering

Reverse engineering is the analysis process that allows to gain a full understanding of a device's design. It has been used to tear down any kind of systems in many different contexts, both for legitimate and also illegal practices. Reverse engineering has been deployed on any sort of objects, from Second World War aircrafts to the smallest most modern integrated circuit, and the motivations have varied from the need to gain knowledge on the enemy's weapon systems, through commercial (and maybe not alwasy so fair) competition to patents infringment or IP stealing in law courts. Many motivations [25] are linked to the need for reverse engineering a product, some of them are collected in table 1. Legally admitted ones range from verification an fault analysis to research and development. The simplest way of getting into a new area of business is buying a product already on the market and tearing it down to understand what parts and what kind of know-how are needed for it to be produced. The way the new product is designed will then state if the reverse engineering step of the development was intended for clonig or counterfeiting or instead was aimed at developing a better and more efficient device. On the other hand, if there is the suspect that a competitor's product has been built by coping explicitly a proprietary design, a reverse engineering firm will for sure find out which patent has been infringed or which IP has been stolen.

Because of the intrinsic ambivalence of RE, legal systems throghout the world have been developed to both protect original designs and so defend innovators' investments and at the same time to give means to second comers to gain a position in the market without wasting too much money on redundant research jobs. The idea is to spur a sort of competition that avoids monopolistic tendencies. US 1984 Semiconductor Chip Protection Act was the first legal measure to protect chip mask layouts from being copied, as a sort of copyright for literary work. However, a reverse engineering privilege was added,

Legal	*Illegal*
Failure Analysis	Fault Injections
Counterfeiting detection	Counterfeiting
IP theft detection	IP piracy
Analysis	Cloning
Hardware Trojan detection	Hardware Trojan implementations
Research	Attacks Development

Table 1: Legal and illegal motivations for RE [25]

allowing the reproduction of the masks for evaluating concepts or techniques embedded and use them in an original work [35]. In this way, the difference between a legitimate and an illegal RE process was stated: a reverse engineer would never copy a system's flaws or weaknesses, but rather analise the device to come up with some optimizations of his own, in his new, original design [35]. Cloning and piracy are contrasted by requiring a certain amount of forward engineering in the design of a new product. In the EU, the legal scope of reverse engineering used to be complex and various due to the many different legal systems of all the member states, which were, moreover, not always up to date. Just to quote an example, a french law from the 70's concerning databases used to forbid reverse engineering since it considered chips as a kind of 'physical' database. There is no need to say that this definition could be feasibile 40 years ago, where the functions of a chip were essentially limited to storing information and not much more, but things have greatly changed since then. EU directive 2009/24/EC can be considered to regulate, in a way, reverse engineering. In this directive, reproduction and translation of computer programs are allowed for the only purpouse of interoperability with other systems. Although masks layout are not explicitly mentioned as in the US, the directive states that in the term 'computer program' also the hardware in which programs are incorporated is included. This may refer to memory blocks storing code and data but also the datapath that executes them. Even

if it is not clear what can be done when it comes specifically to hardware, what cannot be done is indeed specified: all the results of decompilation and reverse cannot be used for any application violating the copyright laws [36]. In other words, extracting the code from a memory is allowed, as well as analysising the encryption circuitry overcoming security countermeasures, while distributing it is illegal and considered, lawfully, piracy. World Intellectual Property Organization, the agency of United Nations specialized in international agreements for intellectual property protection, is more specific as what concerns hardware reverse engineering. As to its legal framework, reverse engineernig is allowed in the reach of a scientific research, or for analysis or teaching [38]. The recent discovery of the disruptive attacks such as Spectre and Meltdown is just one of the many examples of how reverse engineering and security research are important for the development of always more and more reliable systems. Reversing is the first step towards rethinking, reprogramming processes for brand new solutions. The practise of reverse engineering is a way of spreading discoveries and knowledge that has played an important role to innnovation in the past years and for sure will do the same in the years to come.

Reverse engineering analysis can be aimed at retrieving two kinds of information: *technical information* and *patent information*. Technical information reports are usually commissioned by manufacturing companies during product development, strategic decisions taking or benchmarking. The patent information are instead asked by specialised lawyers or intellectual property groups, both the ones inside companies and the ones whose only business model is licensing properietary IPs.

As to semiconductor industry, we can distinguish four categories of reverse engineering [26]:

- Teardowns are the easiest RE analises. The product is only disassembled in order to take photos of boards and make an inventory of components. Resulting bill of material can give an estimation of assembling costs.

- System level analysis consider the whole system, made up of hardware and software is analised. Main board is reversed to annotate all connec-

tions among subsystems to draw a full schematic. Probes can be used to analise signals during operation on buses and chip pins. Software as well is disassembled and reversed to make it human readable and get to know how the system works.

- Circuit extraction or *chip level reverse engineering*, consists in delayering the IC to transistor level and rebuild the whole net of connections though all metal layers, untill the complete netlist is extracted.

- Process analysis focuses on the analysis of the materials the system is made of, in order to understand what technological processes were involved in the manifacturing.

Here we will have a closer look at modern ICs Reverse Engineering.

5.2 IC Reverse Engineering Workflow

Integrated circuit reverse engineering is a complex task made up of different phases, each requiring specific equipment, expertise and a dose of patience. Recent breakthroughs in this field made by Texplained company have brought to the development of an automated tool to facilitate some stages of the process, thus allowing to overcome what was making this kind of analysis technique excessively time consuming and expensive [24].

The whole process can be subdivided into 4 main stages, as in figure 28:

- *Depackaging*: the IC is pulled out of its packaging. This stage is often called *Device Depot*

- *Delayering*: the IC is deprocessed layer by layer with chemical and mechanical etching techniques. *Depackaging* and *Delayering* are the laboratory based sample preparation stages.

- *Imaging*: advanced imaging techniques are used to take high resolution photos of each layer

- *Analysis*: each standard cell, each transistor must be annotated to produce a schematic of the whole netlist. Analysis is performed on the *Analog* circuitry as well as on the *Digital* logic and on *memory blocks*. The flat netlist at transistor level is then translated into a hierarchical one, higlighting functional blocks. At this stage, eventual countermeasures are evaluated and code and data can be dumped from memories.

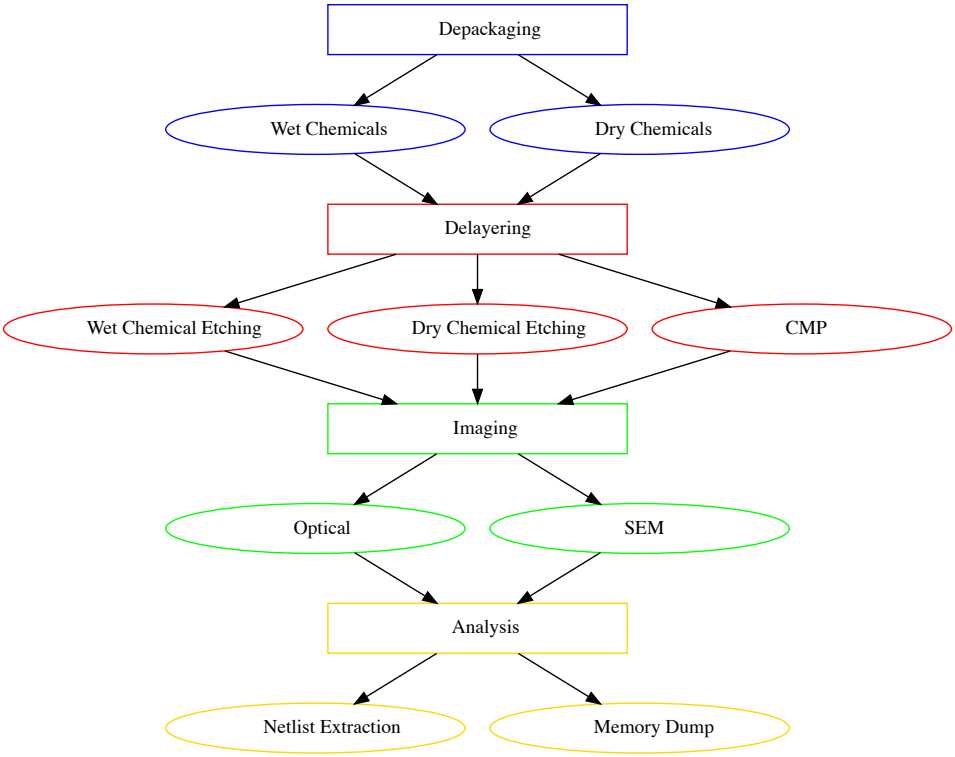


Figure 28: Workflow graph of the RE process

5.3 Depackaging

During depackaging, the package is opened to reach the IC. In case of a *Bare Die Decapsulation* the aim is dissolving away the packaging without harming in any way the die inside. In some cases bonding wires will be destroyed as

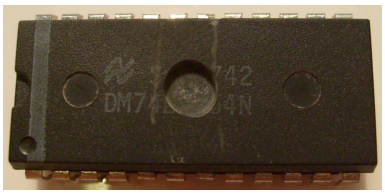
well. Instead, a *Live Decapsulation* enables to dig a hole in the package to dissolve the area covering the to surface of the IC. Top layers are exposed as well as bonding lines. This kind of decapsulation is useful when dealing with SoCs in order to rebuild the connections of the chips inside the same package at the very beginning stage of the reverse process or in case the interest is probing some lines while the chip is working.

The most common packages nowadays are made of plastic. Many different approaches are decriberd in literature to rip off this material. Usually, some acids are used at different temperture, depending on the kind of packaging. Acids should be handled only by experts taking precautionary measures, since hot fuming nitric acid and fluoridric acid are both particularly hazardous and can cause serious burns if they come in contact with skin. All chemical reactions should be carried out in a fume cupboard to prevent fumes from being inhaled.

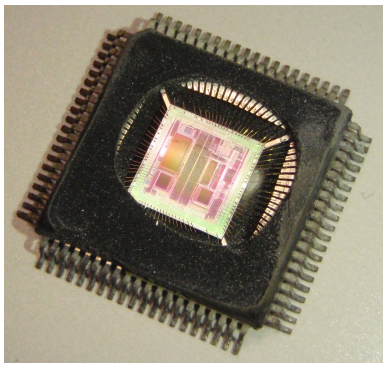
The mainly used wet chemical etchant for removing plastic packages is hot fuming nitric acid, concentrated at least over 95%. For a bare die recovery, it is sufficient to drop into a glass beaker the chip for 3 to 7 minutes to get the plastic package completely removed. The beaker is usually heated to 60°C degrees to speed up the process. Sometimes it could be necessary to repeat this passage to remove the lead frame below the die. The lead frame in fact would prevent backside imagery, which is a fast way to get to the polysilicon layer and so a quick way to get to see memories architecture. After immersion in hot fuming nitric acid the sample needs to be cleaned up, usually done by means of acetone or isopropanole and then water. A ultrasound bath could help in removing residuals. Finally, the exposed die is dried with compressed air. Nitric acid easily attacks copper, but not alluminum and gold. If bonding wires are aluminum or gold made and a full delayering needs to be done, they still need to be detatched from the die. Gold is a particularly tough material that is only attacked by an unstable mixture of nitric acid and hydrochloric acid in a molar ratio 1:3 called Aqua regia. If bonding wires are made of copper, they would disapper during decapsulation. In any case, if there is the need of accessing the chip by using its interface, a wire rebonding would be

necessary: figure 29c shows a bare die rebonded on a new support with gold wires.

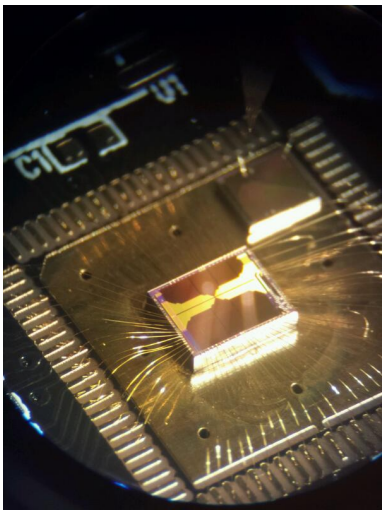
For a *live decapsulation*, plastic package is pre drilled with a milling machine to dig a little circular valley in the area to be exposed as in figure 29a. Manual front milling machines [29] can be used as well as numerical controlled ones, it can be performed mechanically or by means of a laser beam. Some drops of acid are then poured in to remove remaining plastic over the die, as showed in figure 29b. Package should be heated as well to speed up the process. Some drops of acetone stop the reaction when top layers and bonding wires are exposed.



(a) Hole milled in a PDIP plastic package. Source: [31]



(b) Live decapsulation shows bonding wires still attached to the IC. Source: [31]



(c) New bonding wires connected to the die after decapsulation in a rebonding machine. Source: Texplained

Dry etching techniques are essentially based on RIE or plasma etching. High energy oxygen plasma attacks plastic quite well, digging a hole through the package at a hourly rate. The advantage of using automated tools is the periodical precise removal of plastic residuals from the chamber by means of short bursts of nitrogen [29]. However, this technique is not very common for depackaging.

Non selective methods like mechanical techniques or laser ablation were used in the past agsinst old technology nodes, in particular when dealing with metallic or ceramical packages. Metallic DIL and PGA packages for example used to opened by means of a blade applied over the solder joints and then hit with a hammer, or by using a sharp screwdriver to remove metal cover, while ceramic packages were sheared off mechanically under a thermal stress that weakened the glue. Since mechanical procedures are to some extent risky, also time-consuming but extremely cheap polishing with a diamond plate can be used to grind the package and reach the target. Ultrasonic bath is then mandatory to clean up all the residuals. Laser ablation used to be a fast and cheap way of completely uncover a large part of the target while keeping the chip alive. It was used to remove passivation layer and expose top metal layer when a large area, like an EEPROM, needed to be photographed for analysis.A basic form of protection from inspection, for example, used to be made of metal plates covering memories. This makes an attacker lose some time in keeping on hitting with a laser beam the surface untill all metal is gone. Nowadays, most secure microcontroller embed shields as a countermeasure. Exposing passivation with a laser beam would mean just getting to the shield and destroying the shield would mean killing the chip.

Depackaging technique		*Advantages*	*Drawbacks*
Chemical	Wet	High etch rate	Can damage bond wires
	Dry	Good Selectivity	Can result in irregular surfaces
Mechanical	Polishing	Flatness	Not for specific area
	Milling	Easy	High risk to kill the die
	Thermal Shock	Inexpensive	Can damage the die
Laser Ablation		Cheap and Fast	Only for big areas

Table 2: A sum up of some depackaging techniques [25]

5.4 Delayering

Delayering is a destructive process aimed at exposing completely all layers of metal interconnects and expose active areas at the transistors layer, in order for them to be imaged and so to rebuild the netlist. High resolution photos of each layer and sometimes also interlayers are taken. Since photos must be processed by an automated tool, all layers should be exposed uniformly without compromising any feature of the IC. The main challenges in this sense are *repeatability* and *flatness* [28], which allow good quality photos.

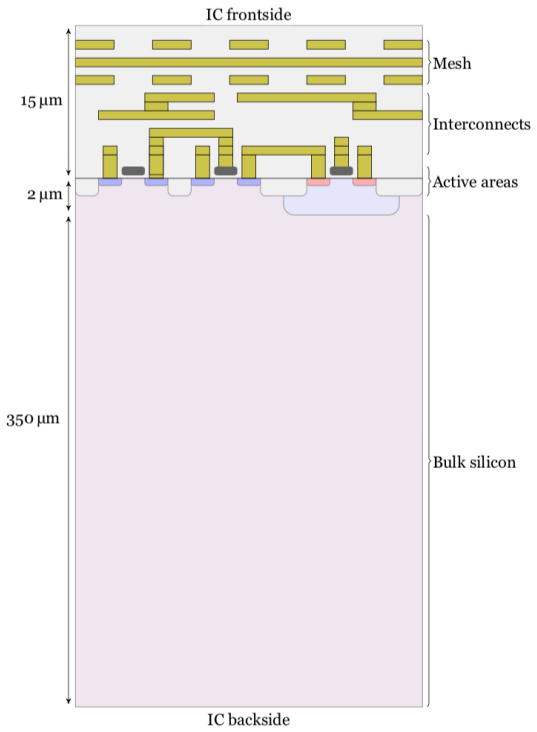


Figure 30: Cross Section view of a standard CMOS process. Metal layers are oriented in alternative directions. Bulk can be thinned during preparation step to get a backside image, which could help identifying important blocks like memories. Source: [30]

Cross section is often prepared as a preliminary step because it allows to retrieve such information about the IC as layer number and thicknesses, structures, different materials in the stack, all quite important to decide which

is the best approach for deprocessing. From cross section, for instance, it is possible to understand that a copper process was used, meaning that the best delayering technique will be a purely mechanical approach.

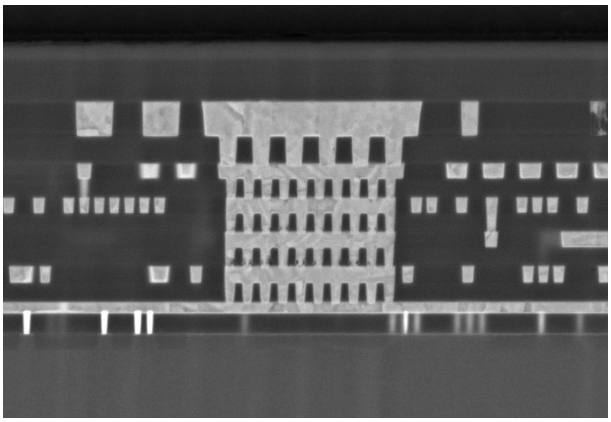


Figure 31: Cross section SEM image showing a copper process, 5 metal layers IC. Tungsten contacts are clearly visible on the very bottom, plugged to transistors. Source: Texplained

Sample preparation for a cross sectioning is divided into four steps: cleaving, polishing, etching and sputtering. Cleaving means just applying a little pressure with a diamond abrasive plate on the die and then force on the cleave: the silicon will split along a crystallographic axis on a pretty straight line. The sample is then polished down to the desired thickness to image the interesting section. Computer aided polishing stations allow to detect automatically the point of interest on the surface and check for progresses during the lapping. This increases the success rate and speeds up the process. Etching can be performed by chemical methods or by ion beam etching. Usually, ion beam etching is preferred since it is difficult to avoid metal lines smearing around contours, which would be a great damage in such a small surface to be imaged. Oxide glass is thinned more then metal lines and VIAs, so that the surface has metal connections exposed. A gold or goldpalladium film is finally sputtered on one side of the sample to avoid charging related problems in SEM images.

As to frontside delayering, different approaches are available, based either on wet chemicals or on dry chemicals and purely mechanical ones. Wet chem-

ical techniques can be used to remove interlayer dielectric as well as metal interconnects. Theese techniques consist essentially in plunging the sample in an acid beaker, using different acids for different materials. Hydrofluoric acid (HF) is the most commonly used to remove silicon oxide, as shown by the reactions

$$SiO_2 + 4HF \rightarrow SiF_4(g) + 2H_2O$$

$$SiO_2 + 6HF \rightarrow H_2SiF_6 + 2H_2O$$

this acid attacks also aluminum, even if the etch in this case proceeds very slowly. The process must be controlled very carefully so that silicon oxide is uniformly removed without damaging too much the aluminum and the layer below, in order not to cause spoils in the photos. Knowledge of the thickness of silicon oxide is mandatory to choose the right etchant with the right etch rate and so avoid defects like edges under-etch or contact holes positions. The reaction is stopped as soon as it reaches the wires and exposes them a little bit.

If we are dealing with an aluminium process, the interconnections are made of aluminium while the vias are made of tungsten. Once metal is exposed, it can be imaged. Aluminum will give a high contrast to SEM with respect to interlayer dielectric, which would be all black in SEM images, while wires are far more clear.

Aluminum can be removed by means of a 65% concentrated solution of phosphoric acid (H_3PO_4), preheated at 50°C degree, which acts very lightly against oxide, or by a mixture of 30% each of hydrochloric acid and sulphuric acid, which does not attack at all oxide. In this way, reaction stops automatically when all aluminum is removed. Once aluminium is removed, trenches are dug where metal used to be, leaving the tungsten vias exposed through them. Now images of vias among layers can be taken. Tungsten acts as an etch stop, simplifying the process since no attention to timing has to be taken care of. Vias are easily recognized since they are the brightest structure in

SEM images, giving the highest contrast with respect to the dark grey silicon dioxide.

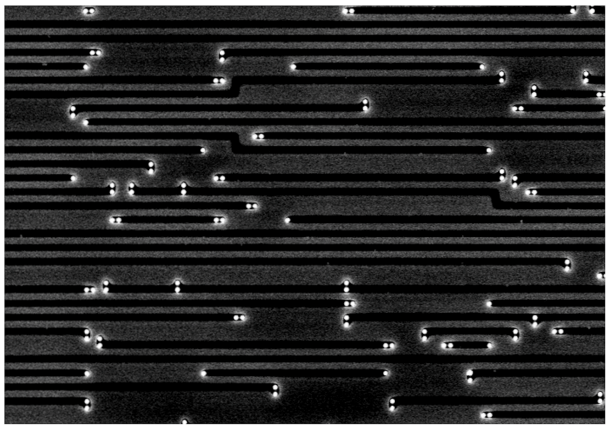


Figure 32: Example of a metal layer image. Metal has been removed to show the vias going to the layer below. Source: Texplained

The same steps are followed for each layer, allowing to take images of both metal interconnects and vias connecting with the lower level. When the active areas are reached, also polysilicon lines can be removed to expose the diffusions. Since doped silicon gives a contrast to SEM similar to that of polysilicon lines, having also diffusion layer imaged may be useful when reversing standard cells.

In reverse engineering sample preparation only one kind of material should be removed at a time, but increasing miniaturization of technology nodes and the introduction of new and complex materials make it difficult to delayer an intergrated circuit by means of wet chemical procedures only. Silicides compound for example cannot be removed by wet chemicals without damaging the rest of the sample. Moreover, getting a good planarity for all the layers by wet etching is quite tricky a task. In fact, since ILD layers are not perfectly regular and flat, the acid would act mostly in the little vales of the glass, resulting in an irregular surface, troublesome to imaging. Wet chemical etching acts isotropically and may not be sufficiently selective as to the material to remove. This would result in smeared metal lines to be seen in the photos, unsuitable for an automated image processing for netlist extraction. Dry etching

in plasma instead allows for either an isotropical or an anisotropical etching and allow for a more selective etch [29]. Dry chemical etching allows for more uniformly flat surfaces to be obtained than wet chemical methods.

This is why wet and dry chemicals are often used together in the delayering stage.

RIE or plasma etching can be used to remove completely passivation until top layer metal is reached and then remove all ILD layers. Silicon nitride added to passivation is a tough material to be dissolved, for which plasma etching with oxygen in addition to fluorine is commonly used. Wet chemicals methods for removing silicon nitride include boiling to 160°C degree sulphuric acid or etching in hydrofluoric acid, both of which attacks quite harshly aluminum and metal in general.

Oxide glass and ILD is removed by dry etching until the metal is exposed and imaged. Different gases are introduced in the reaction chamber to attack selectively different materials, from oxide to nitride and silicides. When metal is exposed, wet chemicals are used to safely and selectively remove it. Dry ethcing can be used to remove also metal, as well. Table 3 shows some of the most used gases for some common material for modern CMOS process.

$Material$	$EtchingGas$
Si	CF_4, C_2F_6, C_3F_6
Si_3N_4	CF_4
Al	$CCl_4(+Ar), BCl_3$
Mo, W	CF_4
Ti	CCl_4
$MoSi_2$	SF_6
$TiSi_2$	CCl_4
WSi_2	CF_4

Table 3: Gases used for different materials etching. Fluorine gases are preferred since they are not corrosive and cause no damage to equipment [29]

The main problems to be dealt with during dry etching are *underetching* and *overetching*. As to the first, in some spots the current metal layer is not exposed so that it is not possible to rebuild the network of connections for that layer; as to the latter instead the metal layer is overexposed, with the etching reaching the next metal layer, showing in the pictures a superimposition of connections that makes no sense at all.

Modern ICs are made with a copper process. Copper interconnects allow for smaller geometries to be reached solving problems posed by parasitics parameters growing with scaling. Moreover, in a way, this process provide a higher level of security since it is more difficult to delayer. In copper process, in fact, both connections and vias are made of copper by means of dual damascene technique. In this case, nitric acid would etch all the way down through the dual damascene and stop only at the tungsten contacts on polysilicon layer, which it does not etch at all.

In general, controlling all the chemicals needed has became complex, time consuming and risky, resulting in poor reliability and repeatibility of the process. The tendency, for modern processes, is to move to mechanical delayering. During mechanical delayering the sample is glued on a carrier that is slightly pressed on a rotating plate. Different techniques are available to stick the sample on the carrier, usually a wax glue is adopted due to its good behaviour with temperature. During *Polishing*, diamond abrasive plates with different thickness according to the desired resolution are used to wear thin the layer until it is completely removed. There are many mechanical polishers and techniques that can be used. Manual systems require a great expertise to get good results and however do not allow for high precision and repeatability. When studying a device for reverse engineering every sample is unique and the process must have a high degree of control. Computer controlled polishing systems, like the one showed in figure 33 instead allow for parameters like tilt angles and speed to be controlled and adjusted [28].

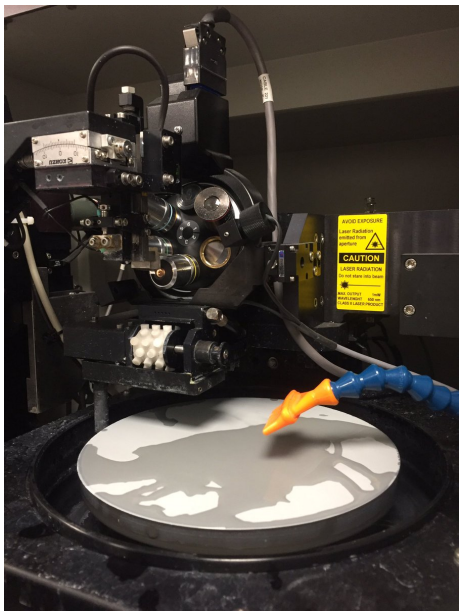


Figure 33: Diamond abrasive rotating plate is sprinkled with some deionized water to cool down the temperature on the the chip while running. An optical system allows for the user to see the proceedings of the job on the sample inside the pressure vessel. Source: Texplained

The main concerns for this technique is the impossibility to expose completely a whole layer due to side effect. Polishing in fact acts the most on the edges, resulting in evident rounding on the angles, which may be unacceptable for netlist extraction.

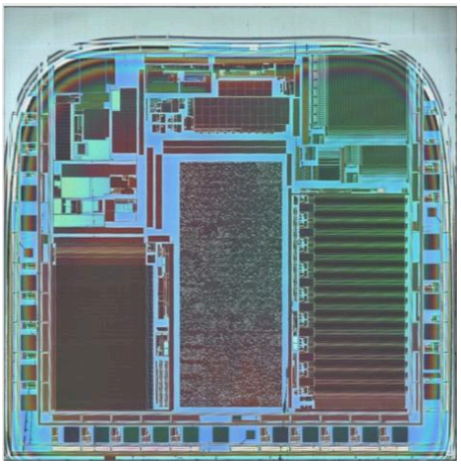


Figure 34: Evident side effect due to polishing. Edges are completely worn out. Source: [28]

The innovative method proposed in [28] totally overcomes side effect and allow for a 100% rate success, by using a silicon sacrifice carefully aligned to the leading edge of the sample and a new, ad hoc conceived recipe, reported in table 4.

Step	*Abrasive*	*Force*	*Rotation*	*Speed*
1	$6.00\mu m$	$200g$	CCW	$100rpm$
2	$1.00\mu m$	$200g$	CCW	$60rpm$
3	$0.50\mu m$	$200g$	CCW	$60rpm$
4	$0.10\mu m$	$200g$	CCW	$60rpm$

Table 4: Special recipe for polishing with no side effects [28]. Finer and finer abrasives at each stage are used to polish smaller and smaller geometries surfaces

With this method, the sample can be succesfully delayered to expose completely each layer, as showed in figure 35.

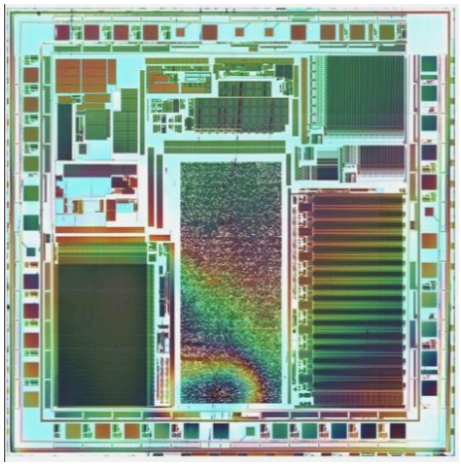


Figure 35: No edge has been worn out, full connections network for each layer can be retrieved. Source: [28]

5.5 Imaging

Imaging represent the data acquisition phase during the RE process. Main imaging techniques used nowadays are *Optical Microscopy* and *Scanning Electron Microscopy*.

The simplest way to inspect is using an optical microscope. Optical microscopes can reach magnification up to 2000x, do not require a high expertise and working station are quite quickly set up. An on board CCD camera allows for taking high resolution images of small portions of the samples, that are sticked together as tiles of a mosaic to produce a full, coloured reproduction of the IC under analysis. The biggest limitation of optical microscopes is resolution. Optical resolution is estimated as $\lambda/2NA$, where NA is the Numerical Aperture of the lenses. However powerful, optical microscopes can reach up to 220, 250 nm resolution and could be used for reverse engineering down to 0.18μm technology nodes [4].

Nowadays, optical images are used for quick eagle eye view of the IC as a whole, for identifying main structures and high level system architecture. Big strustures like SRAM banks, ROM and Flash memory blocks are easily recognized as well as main digital logic section and big analog blocks. This

kind of information are useful for tracing a quick identikit of the target and may represent an initial, gross verification if information given in datasheet are actually true, which is not so expected as it may seems. SEM images will then be used to actually investigate in detail all interesting system's features.

The Scanning Electron Microscope is by far the most important tool for reverse engineering imagery. In SEM an electron beam is used instead of light as illumination source and is rastered over the surface to be imaged. Signals coming from the interactions between electrons and the sample are collected by dedicated detectors to form the image are generated in different areas of the interaction volume as reported in figure 37.

- *Backscattered Electrons* are electrons which escape the surface after one or more scattering events. These electrons are deflected to source by sample's atomic nuclei with a wide angle change. Typically their energy is greater than 50 eV. Elements with a higher atomic number have a higher number of positive charges, thus reflect with better efficiency incident electrons, causing the resulting backscattered signal to be higher. As a result, contrast in SEM images will be higher. Heavy elements as tungsten and gold appear extremely clear with respect to lighter elements. Due to their high energy, BSE are not rapidly adsorbed but are produced in a larger area below the surface. This allow BSE to carry information regarding a wider region but cause their resolution to be quite high, equal to about 1μm. BSE can provide both compositional and topographic information in the SEM [32].

- *Secondary Electrons* are the electrons belonging to external energetic levels of the sample, emitted when the primary beam hit the sample causing its ionization. Typically low energetic, 35 eV, they provide information from the area pretty close to the beam spot with a good resolution, 10 nm or even better. Secondary electrons are used to reveal surface texture and roughness [32].

- *Characteristic X-Rays* are emitted when the electron beam removes an

electron from an inner energetic level of the sample, causing a higher energy electron to fill the hole releasing energy. Analysis of characteristic X-Rays reveals the chemical nature of the sample and its composition, like when diffusion areas are imaged as in figure 36 [32].

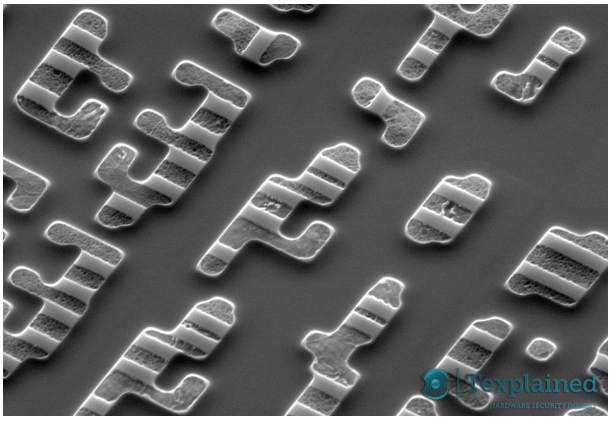


Figure 36: Diffusion areas marks on bulk. Polysilicon lines signs are still visible after removal

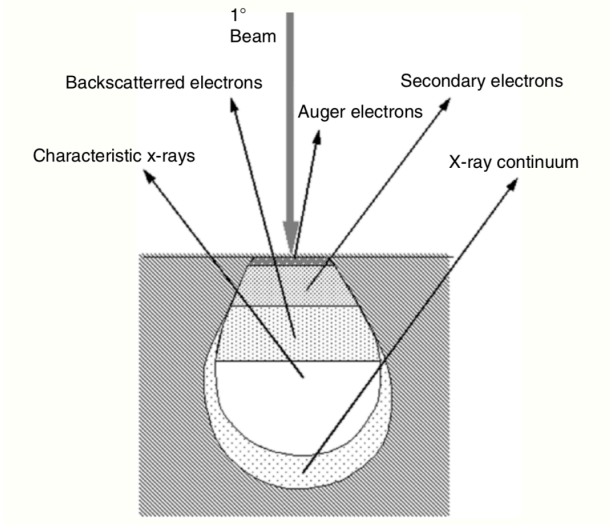
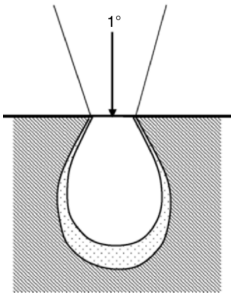
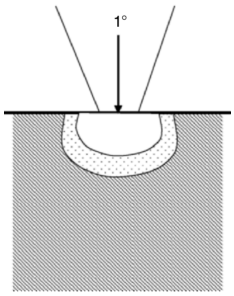


Figure 37: Signals generated by electron beam and relative generation areas. The whole volume is referred to as Interaction Volume and its size depends on beam's acceleration voltage. Source: [32]



(a) Higher acceleration voltage means a bigger interaction volume. Images are brighter but with less resolution and more noise due to information on material beneath the surface. Source: [32]



(b) lower acceleration voltage means a smaller interaction volume. Images have a better resolution but contrast is lower. Source: [32]

A general diagram of the instrument is reported in figure 39

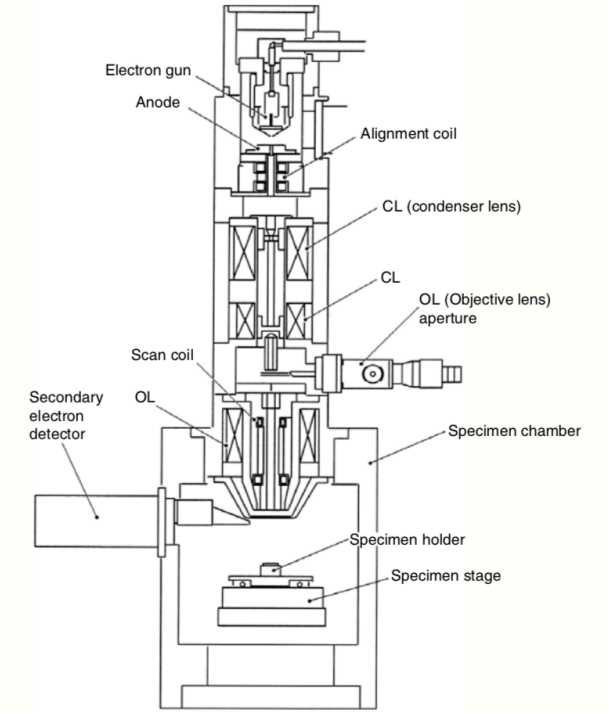


Figure 39: Schematic diagram of an SEM instrument. Source: [32]

The **electron gun** is where free electrons are accelerated in a tungsten filament at 3000 K degrees toward an **anode** that is adjustable tipically from 200 V to 30kV; the beam is then focused until untill it reaches up to 1/1000 of its original size, determining the intensity of the beam on the sample while passing through the **condenser**; one or more **apertures** are used with electromagnetic lenses control the diameter of the spot, affecting parameters like resolution and depth of field: the smaller the spot size, the smaller the resolution, at the cost of brightness; **scan coils** deflect the beam to make it run accross the surface while a dedicated system called stigmator correct for astigmatism and beam's aberrations thanks to the **objective lense**; finally, at the bottom of the electron column, the sample in the **specimen chamber** is hold firmly on a movable stage that can be moved in the vertical direction and on the horizontal plane, rotate and tilt to select the field of view and adjust resolution. All the electron column must be kept at high vaccum, around 10^{-3}, 10^{-4} Pa, to prevent air particles and dust from interfering with the electron beam.

When dealing with non conductive material, SEM is affected by issues related to charging. Electrons hitting the surface are collected on the surface, accumulating a high negative charge that cause a negative potential. Electron scanning the surface are deflected by this potential, resulting in a distorted image. Over a certain value of the potential, electrons are totally deflected and the image are completely black. There are essentially two issues relating to charging. One is to have the sample physically grounded to the instrument. To do this, a silver paint is used to connect the edge of the sample to the SEM stage via the sample stub. However, the surface of the sample is still non or only partially conductive. To overcome this, a very thin (a few nms at most) layer of either gold or a goldpaladium mix is sputtered. Because this thin coating is very broad, it also connects the surface of the sample to the previously used silver paint and thus everything is now conductive.

5.6 Analysis

Analysis takes place on the high resolution images that have been taken for the chip. SEM images need to be corrected for distortion and stitched together to create a full image of the whole layer, for all layers. After that, layers are aligned in a stack and connections are rebuilt, from substrate to top metal, in order to extract the netlist. Analysis is made to acquire knowledge on the IC and thus evaluate how difficult would it be to perform attacks on it. Usually, three main blocks are recognizable inside an integrated circuit, all are interesting for reverse analysis:

- **Analog Circuitry** since some analog structures can be used or abused during attacks, it is important to evaluate how difficult is to find them and above all to get access to them;
- **Digital Logic** performing all state machines, instructions, and algorithms the device is supposed to, represent the main business of the analysis;
- **Memory Blocks** storing data and code, are the main target of attacks.

5.6.1 Analog Circuitry

Analog circuitry plays a crucial role in any IC. Usually it is more difficult to reverse than digital logic since manifacturer can play many tricks on it. Manifacturers can implement functional elements their own way and often solutions adopted are not at all the same as the one in books. After a schematic is drawn, it will take some time to understand the function and the operation of an analog circuit since transistor are not just used as switches but their behaviour depends on many factors.

Analog structures give hints on the function of the many different blocks integrated in the chip. Figure 40 shows a memory array with some analog circuitry on the left side. Analog circuitry can be recognized as it quite always embeds capacitors, which are implemented as big, polysilicon plates easiliy

recognizable from a substrate image. The many capacitors near a memory array, organized following a precise pattern could be a charge pump, meaning that what we are dealing with is either an EEPROM or a Flash memory block. EEPROM and Flash, in fact, need voltages higher than the power supply for their programming and erasing operations, which can be provided by a charge pump circuit, essentially a chain of capacitors charged during opposite phases.

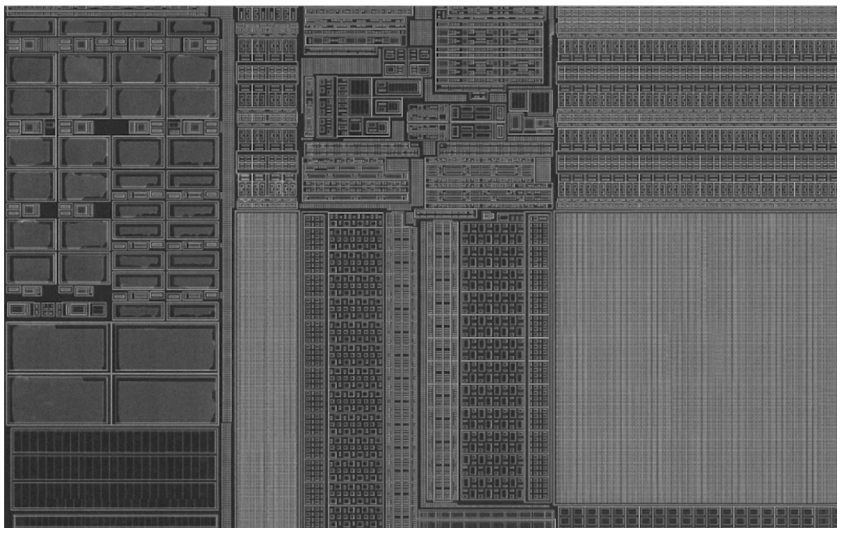


Figure 40: Set of capacitors close to a memory block. Source: Texplained

The whole control of the IC operation flow is dictated by the clock signal, which is created and modified by an analog structure. Following the clock lead to any integrated subsystem and taking over its management can give illimitate power over the chip. However complex could be its management unit, a clock is always generated essentially by one or more RC oscillators, which are quite big and easily recognizable structures made up by passive components.

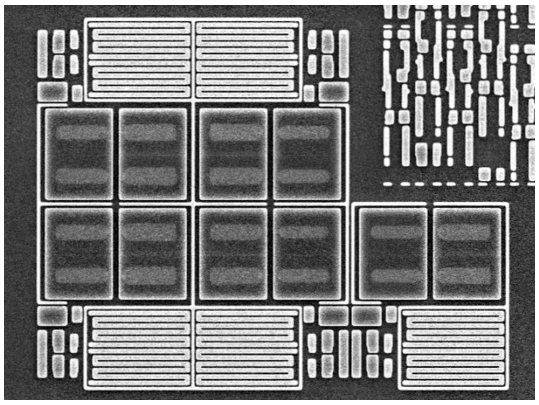


Figure 41: An integrated RC oscillator. Source: Texplained

Figure 41 shows a substrate image of an RC oscillator. The resistor is realized by means of wavy polysilicon lines, bent on themselves many times to get the longest in the smallest area possible. The big square structures are plates of a capacitor, that summed together in a parallel connection make the capacitance value needed for the oscillation to take place at the desired frequency.

Taking control over a clock RC oscilator would give the possibility to disable it or modifying its behaviour, thanks to an access pad placed by means of an FIB edit. A system that features different clock signals for different submodules would be particulalry harmed by this. Let's say the clock for the crypto engine is separated, found and disabled: the entire IC would be left without any means for authenticating access rights, letting the attacker in with no additional checks, for example. In secure devices, RC oscillators can also be used in Random Number Generation units. Good randomness can be obtained by using two RCO, one running at high speed and a slow one, that triggers measurements on the fast oscillator. RNG is used to generate session keys that are scrambled with seceret passwords during authentication. Abusing an RNG would allow to retrieve passwords or prevent the circuit from even using them at all, letting the attacker gain access by sending any value for verification.

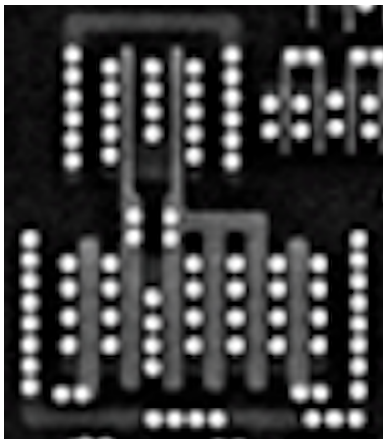


Figure 42: A sense amplifier

Common structures that are almost always found in memory output paths are sense amplifiers, like the one shown in figure 42. Sense amplifiers speed up the read operation by comparing the value of the bitline to a precharged line. This step of the reverse analysis process is made by hypothesis and assumptions, which can be confirmed or not. Usually one has an idea on what a structure may be, tries to reverse and see if own's intuition was right. If there is the suspect that a particular line is actually a bitline, the presence of a sense amplifier will confirm the hypothesis.

5.6.2 Digital Logic

Digital logic analysis takes place mostly, but not exclusively, at the level of standard cell. Main core and logic units are mostly implemented as standard cells connected together and placed among power supply rails, disposed either in raws or columns. Standard cells analysis starts with a collection of SEM images digital section at polysilicon layer, where polysilicon lines, diffusion areas and contacts are visible. Here, as shown in figure 43, the relative position of all transistors and their type can be found, but not their relative interconnections. Once identified, the standard cell is tagged with a name, associating its unique ID to its specific pattern of vias.

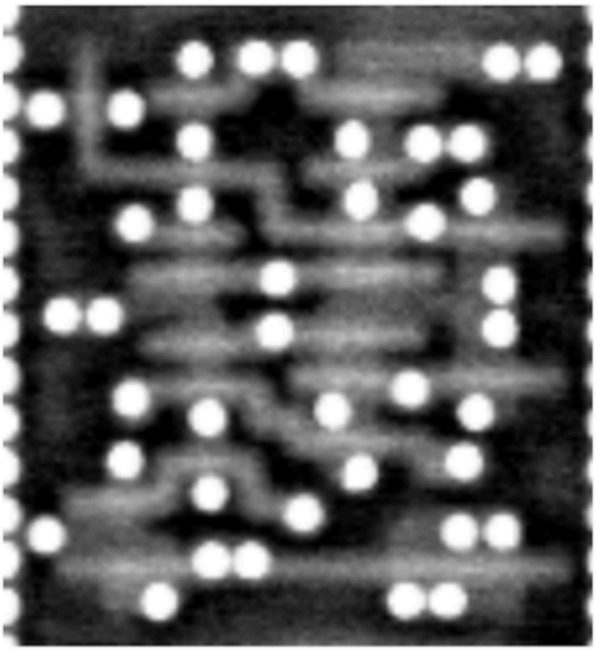


Figure 43: A standard cell is identified at poly layer. It is easier to find the boundaries of an instance on poly layer than at metal 1, where most of the times cells are almost unrecognizable. Source: Texplained

The very same pattern of vias is then searched for in a new set of SEM images, this time regarding Metal 1 layer as in figure 44. Metal 1 has to be deprocessed and exposed so as the SEM is able to take images of both metal lines and vias going down to the substrate, contacts indeed. The same pattern of vias of the just identified standard cell can then be found in the Metal 1 images, so as to retrieve cell's inputs and outputs, internal signals and connections to power voltage supply and ground.

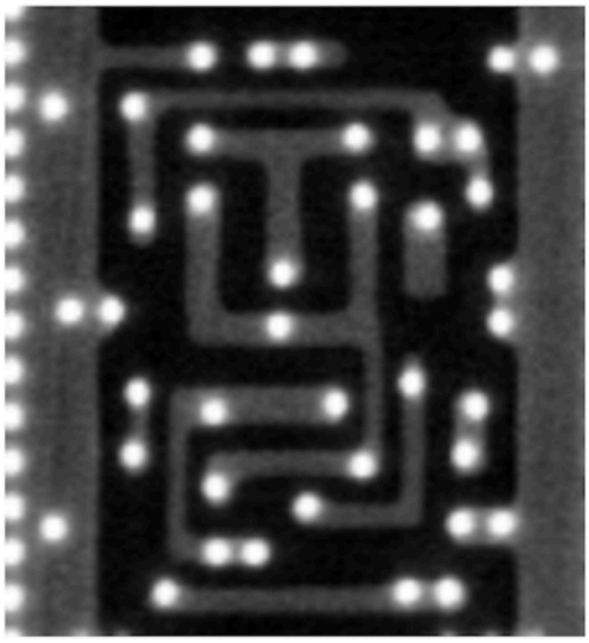


Figure 44: The same cell is found on Metal 1 layer. Source: Texplained

At this point, the two images can be superimposed to draw a view of the stick diagram of the cell. All vias are given a name, in this case a number for internal signals and a letter for inputs. Output is identified as that connection that links together n and p diffusion areas and nothing else. A transistor is created any time a polysilicon line is traced on a diffusion area, having its Source on one side and Drain on the other. As can be seen in image 45, the common tecnique of sharing active areas among neighbouring transistors has been put in place here to save area.

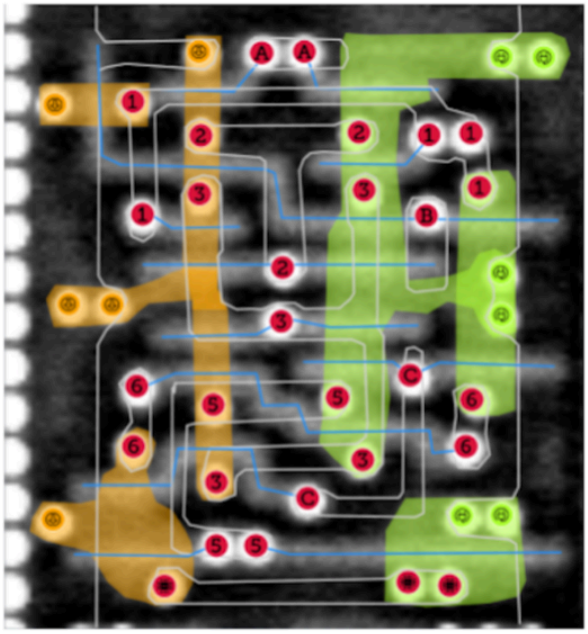


Figure 45: M1 connections are drawn over the image of poly layer, to give a specific function to each of the vias. Source: Texplained

Finally, from the stick diagram of the cell the schematic view in figure 46 can be drawn. Schematic's truth table is then filled and covered to discover the function of the cell. This one, for example, is a three input XOR gate. XOR gates are widely used in secure devices as part of the encryption circuitry, since, given a value A and a key K

$$A \oplus K \oplus K = A$$

meaning that it allows to scramble and descramble any value with a key. Of course, encryption circuits are far more complicated than that, but anyway, whenever a bunch of XOR gates are found in a specifica area, it means that something related to security is going on there.

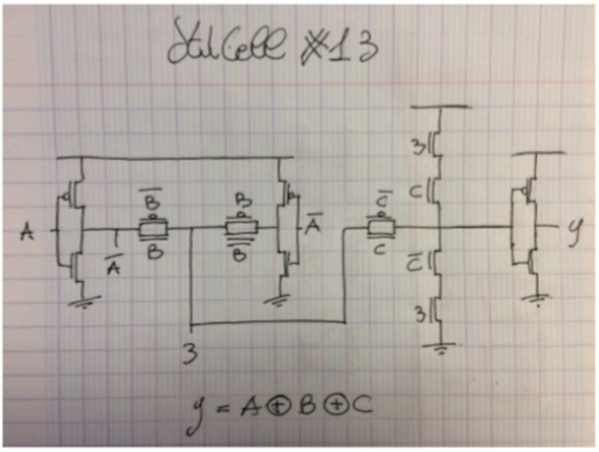


Figure 46: Diagram of the cell traced with pen and paper. This XOR function using transmission gates allow to save silicon area. Source: Texplained

Standard cells are organized in catalogs, designed by the manifacturer for a specfic application according to a set of rules depending on the technology node. This means that the same catalog can be higly probably used to extract the netlist of another chip of the same manifacturer belonging to the same technology node. Usually catalogs are made of a number of different cells ranging from 50 to 200. Actual size depend rather on the level of optimazion of the design than on the dimension of the IC. It can happen for example the very small chips based purely on state machines embed however a huge number of standard cells due to the high level of optimazion that the logic went through to reach strict requirements of power and speed.

Once the cells have all been found and reversed, the interconnects layers are analysed. All lines and vias are identified. After all images of all layers have been processed and stitched together, the resulting images of the entire layers are aligned among them, rebulding the stack from top metal to substrate.

5.6.3 Memory Blocks

Memory blocks are the target of most attacks aiming at extracting the code or the secret data stored inside. A full memory content extraction is called **dump**. Many security countermeasures can be put in place to prevent memory content

from being extracted. Secure devices often embed a memory management unit in the core that checks for access rights for both writing and reading. These units can protect the memory content by, for instance, allowing access only when the right passwords is provided and block the device in case an illegitimate access is attempted.

In the secure microcontroller analised in this thesis work, the code within the Non Volatile Memory is encrypted. A dedicated decryption ciruitry is embedded in the core in order to descramble the instructions before the execution. In this way, even if the NVM block is abused and content extracted, it still would be useless without the decryption algorithm. Moreover, another security measure was taken against code extraction by scrambling the addresses. Addresses are not linearly organized but are scrambled, so that bits and bytes are not written in contiguous cells but distributed according to a particular scheme. Once again, simply extracting the bits would lead nowhere: without the right order given by address descrambling, it would be completely nonsense.

Full reverse engineering analysis of both addresses and decryption circuitry has been performed to be able to read the data in the memory. The ROM dump workflow followed is reported schematically in figure 47.

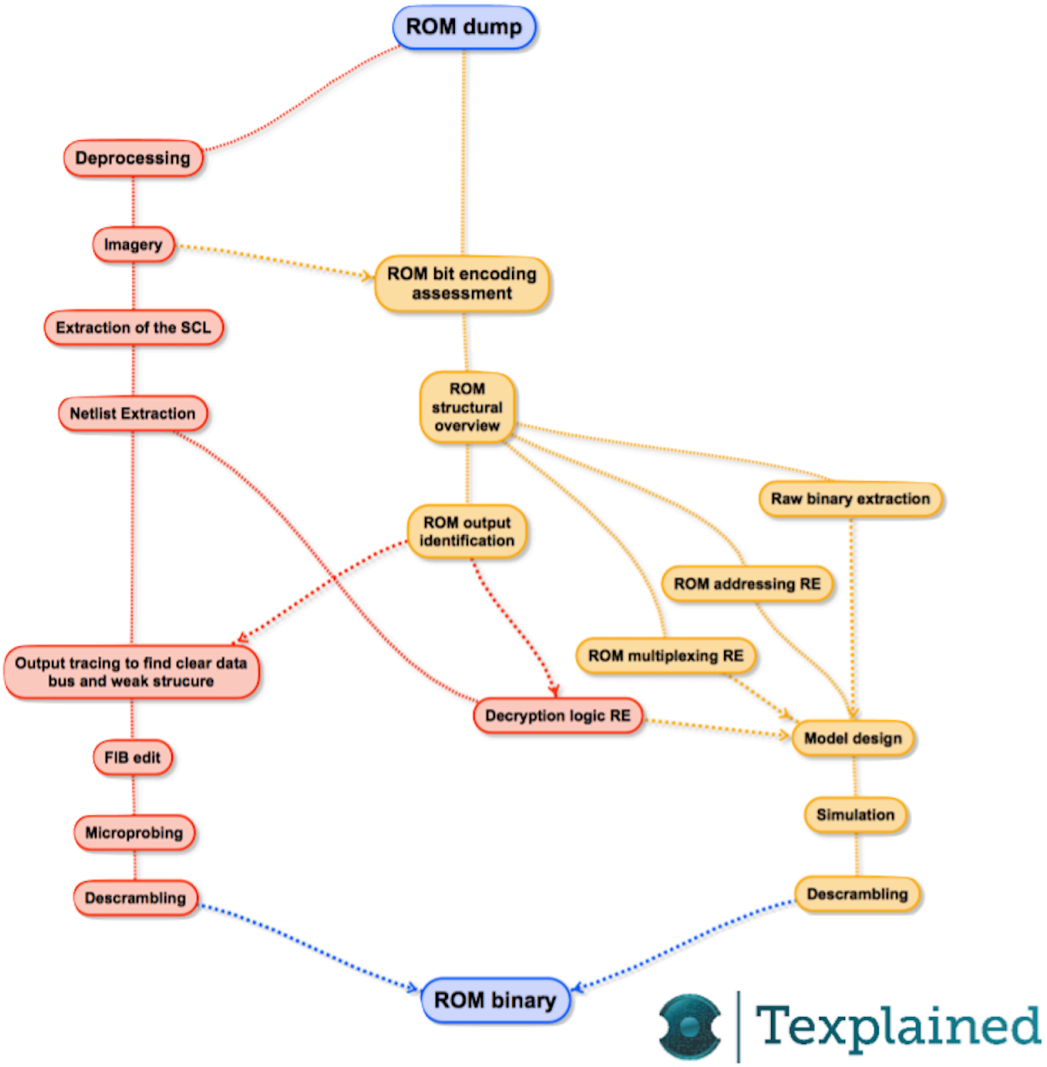


Figure 47: ROM Dump flow chart. Source: Texplained

6 ROM Dump

In this section, the main results of the analysis conducted on the higly secure microcontroller audited will be discussed.

The package of the chip was made of plastic, so it was dissolved away by means of wet etching in nitric acid. Interconnects layer and vias were made with a copper process, so a fully mechanical delayering was performed, using

a recipe similar to the one exposed in table 4. After that all standard cells in the catalog have been identified and reversed, the ROM memory was taken into consideration.

The bits are encoded on Metal 1 through contacts going up from polysilicon layer to the bitlines. Figure 48 shows on the left the grey polysilicon wordlines with VIAs regularly spaced above and below.

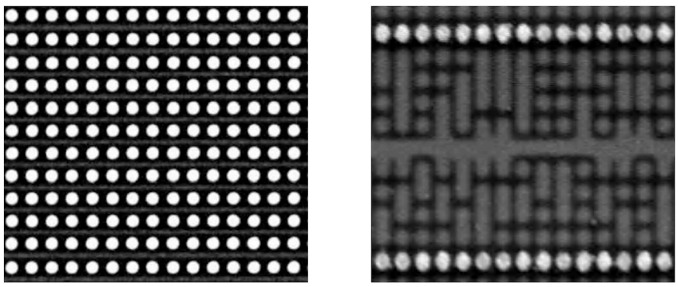


Figure 48: Polysilicon wordlines and contacts on Metal 1 layer. Source: Texplained

The schematic representation of the cells is showed in image 49. Wordlines are grouped by blocks of 5 nMOS transistors, linked together in series and connected to the same bitline thanks to an additional pass transistors. The pass transistor enabling the connection on the bitline is driven by separate control lines. This memory block is therefore a NAND ROM.

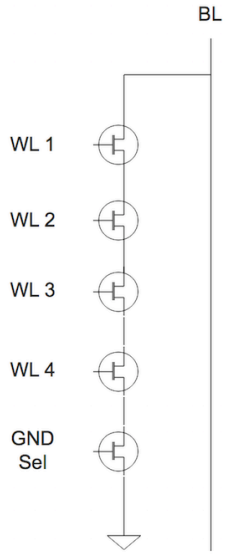


Figure 49: NAND cell schematic. Source: Texplained

Contacts on poly are going up to Metal 1. Here, bits are encoded: where a transistor is shortcircuited, current will pass when its wordline its actived, thus driving the bitline to 1. By knowing this, it is possible to automatically process all SEM images of the ROM to extract the raw bits. The result of this process on one picture is reported in figure 50.

Figure 50: An SEM image processed for raw bits extraction. 1s and 0s written in the memory are turned into white and black spots. Source: Texplained

When all SEM images are processed and turned into a low noise encoding, they need to be stitched together. A script turns the QR code like images into tables of bits, which are easily correlated and overlapped together by a custom software tool.

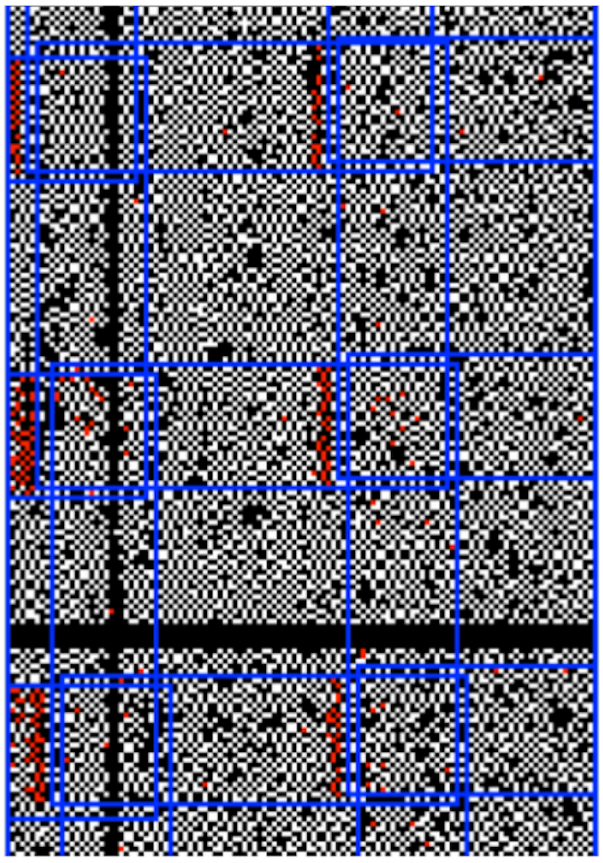


Figure 51: Low noise images are easier and faster to correlate. Source: Texplained

Wordlines are driven from the addressing stage, reported in figure 52. The memory is divided in blocks of wordlines, driving a group of transistors connected in series to the same bitline. Different lines are used to select the desired block and the particular wordline whithin the block.

During analysis, a gap in the address space was found. Wordlines number is less then all the possible addresses that can be covered by address lines: there is a small area in the middle of the memory which is addressable but not

reachable. From the size of the code stored in there and by knowing that the secure chip embeds a crypto core, it is possible to make the assumption that this code chunk is made of crypto libraries. This area is separated from the rest of the accessible code by some empty lines: in this way, even if a the next instruction pointer is abused or undergoes an overflow, it cannot end up here.

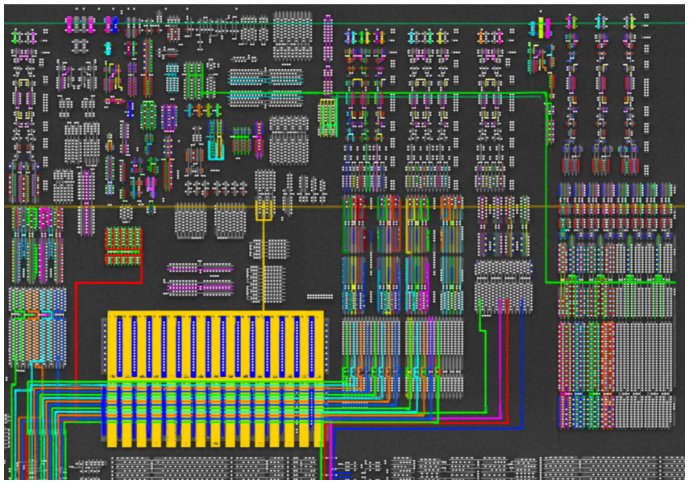


Figure 52: Addressing circuitry for wordlines selection. Source: Texplained

Figure 53 shows the schematic diagram resulting from the analysis of the addressing driving wordline selection. The whole schematic view of all wordlines is reported in figure 54

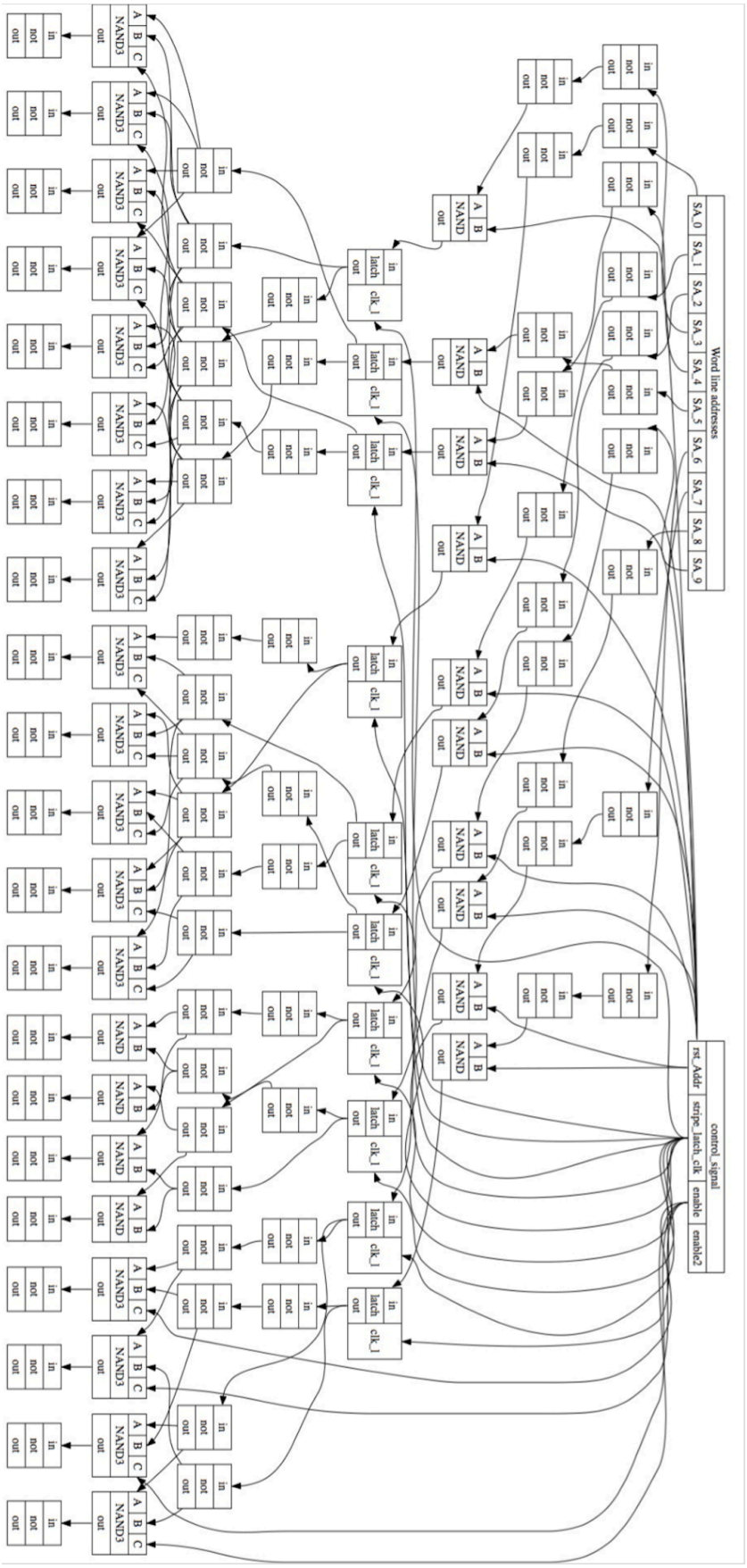


Figure 53: Schematic view of the block of wordline selection. Source: Texplained

Figure 54: Schematic view of all the wordline blocks whithin the memory. Source: Texplained

The python script 1 shows how the addressing circuitry has been emulated, generating the corresponding stripe of wordlines selected and the pointer to the position of the wordline from linearly ordered addresses. Multiplexing logic instead is simulated in script 3, that indicates the pointer to the selected column of the memory for all the outputs of the bitlines.

The decryption circuitry is integrated in the core, between bitlines outputs and the instruction register. Inputs of the decryption stage are memory outputs, address lines and some secret keys. Adding addresses into the encryption allows to reach a fine grain encoding, particularly difficult to break without a

full reverse analysis. Keys are stored somewhere else in the IC, and can be hard coded in the core or uploaded in a special register at start up. Decryption circuitry schematic diagram is shown in figure 55. The first raws are essentially made of XOR gates. In the last stage there are some flip flops, the instruction register.

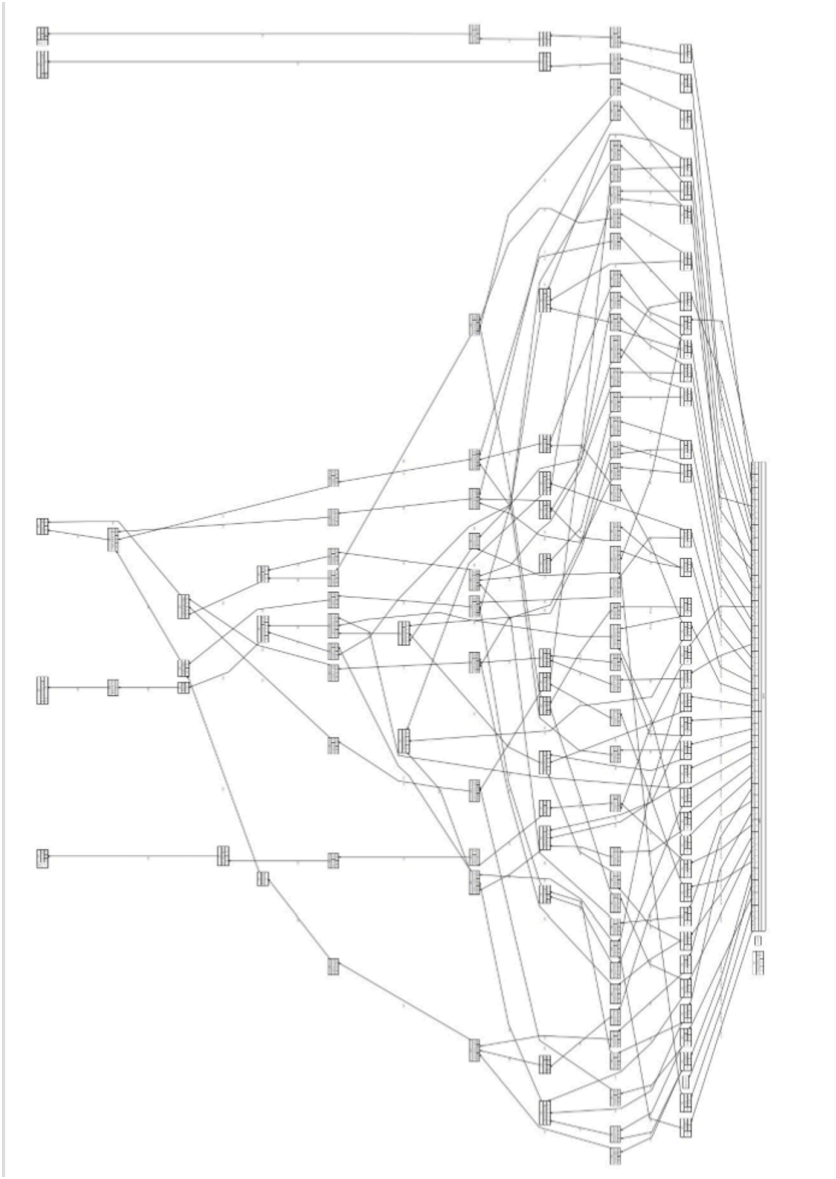


Figure 55: Decryption circuitry is organized in stages made up of similar standard cells. Source: Texplained

7 Risk Assessment

At this point, once that all about memory architecture and addressing scheme is known it is possible to proceed in 2 different ways towards full memory content retrieval:

- Analitcal Extraction, a reverse engineering based dump technique
- Linear Code Extraction, an older, fully invasive memory read out technique

7.0.1 Analitical Extraction

Memory content is already imaged in a low noise format. What we have at this point is not yet the code itslef. Data stored in memory are encrypted and scrambled accross the entire space so that all the 0s and 1s imaged make no sense at all. Code must be extracted from these raw data.

Bits must be ordered and decrypted. Addressing scheme is translated into a script, in this case a Python one, due to its portability, that emulates the just reversed hardware address scrambling. Inputs to this script are linearly ordered addresses, as the ones the microcontroller would ask for to the memory, and the outputs are generated according to the way the address encoder is working in the device. The outputs are used as pointers to the position in the sea of bits extracted from the memory where the corresponding piece of information is stored. In this way, it is possible to order all code inside the memory as it is meant to be read. Data at this point are ordered but not yet decrypted. Another Python script is prepared to emulate the behaviour of the decryption circuitry. Inputs for this script are all the outputs of the previous one, one raw after the other. At the end, the final result shows the whole content of the memory in plain text, ordered linearly as shown in figure 56.

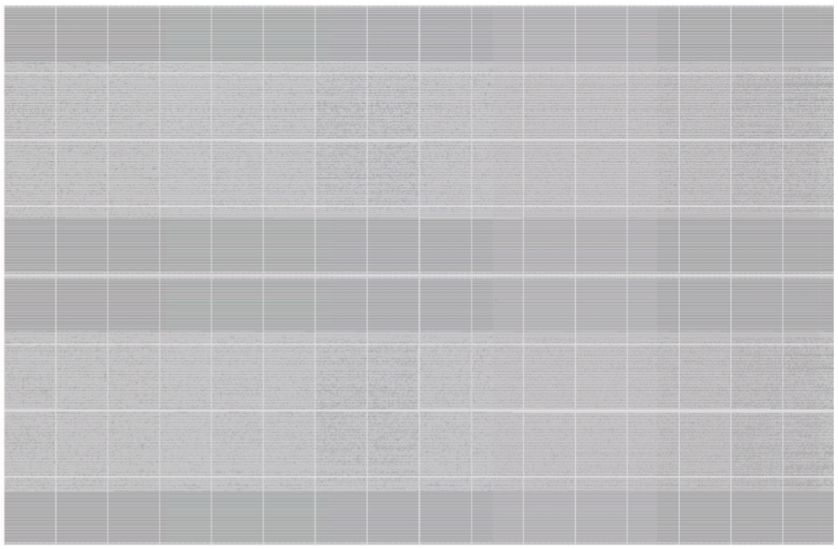


Figure 56: ROM bits extracted and decrypted. Source: Texplained

It is sufficient to find the right disassembler tool to transform these binaries into the assembly instructions written in the ROM. ROM memories often store bootloaders, that contain some test routines to check the working state of eventual Flash and EEPROM on the chip. These routines could be used to extract all code within these memory blocks.

7.0.2 Linear Code Extraction

Linear Code Extraction or *Memory Linearization* is a quite old tecnique invented to perform smart card memory read out [21]. This fully invasive attack allows to dump all memory content by making the microprocessor read it out while probing the bus. LCE is used mainly on NVM memory blocks containing boot code. If the target IC embeds both a ROM and a Flash it is better to start attacking the ROM, since it is more probable that boot code is written there. Test routines for Flash read out found in a system configuration area whithin the ROM can be exploited to get the Flash dumped simply by asking the chip to via SPI, for instance. The name *linear* stands for the way memory is read: at the end of the attack we will get full memory content in plain text and ordered by increasing address. In our case, the target is a ROM memory.

Let us divide instructions into two main groups: *sequential* instructions, that do not cause any branch and let the program counter call for the next instrusction during execution, and *jump* instructions, that cause a branch into a different area of the memory. The latter kind of instructions must be prevented to perform a LCE since a jump would break the linearity.

The first thing we need to do is find where to probe the bus, i.e. how to find the instruction register. In fact, even if the memory is encrypted, it is sufficient to probe beyond decryption circuitry to read the content in plain text. Boarders of decryption circuitry ending in the bus can be spotted by making 2 considerations: first, XORs gate are a very good tell of encryption; second, very often, especially when different types of memory are integrated on the IC, multiplexers are used to switch among different memory outputs after encryption.

Then, we need to find a way to trick the execution of jumps. Let us assume that, at power-on, the first instruction the chip executes is a sequential one. At this moment, the IR could be zeroed by a reset to be sure to set the microcontroller into a known state. Since the first instruction is not causing a jump, the next instruction the address register is calling is written in the position next to the current instruction. If the instruction register is frozen in this state, it will keep on causing the CPU to call the next instruction, reading all the memory one instruction at a time in a linear order. The best method to achieve this is to manipulate the ENABLE control signal of the IR, so as to prevent from latching another instruction. Controlling the EN signal gives us the power to decide when the extraction must be linear. The chip could in fact start executing code from a location different from address 0: when the bottom is reached, how could we get back to the beginning without a jump? Letting the IR latch a jump instruction allow to move through the address space as we like. In some cases, when there is no jump to address 0, the lowest address jump is taken and then the few bits separating from 0 are modified by means of an FIB edit.

Only a few microprobing needles are needed to perform this powerful attack. One needle is used to control the enable signal of the instruction register;

a second needle is set on the clock, as read control to synchronize the acquisitions: each time the clock switches, new data are moving on the bus; a third needle probes the lines of the bus one at a time; a fourth needle could be additionally fixed on the first bus line, introducing extra redundancy to the acquisitions.

7.0.3 Comparison

The cost of an LCE attack in terms of equipement is for sure higher than the analytical memory extraction's one. SEM pictures of all layers of the IC must be prepared and studied to find the exact position of the IR, which can be performed with common, commercially available softwares. The time needed for this preparatory step can range from 1 week to 1 month, which is however an acceptable unit of time since usually, in risk assessments, time is a resource considered as infinite for attackers. FIB edit is needed to build pads on specific locations in the IC for the probes to sense bus lines and control the ENABLE signal. Rather than buying all the expensisive machinery needed, it is more convenient to rent everything from laboratory facilities. Costs for rent are about 1,5k dollars per day. Standard time needed for the FIB edit and microprobing part of an LCE on a medium complexity chip is about 2 weeks, for a total of less than 20k dollars.

The analytical ROM dump instead does not require further access to the sample after deprocessing, which could also be performed by a specialized laboratory without the need to prepare anything on his own. A medium size standard cell catalog is filled and reversed in about 2 weeks, while other 2 weeks are needed to export all interconnection layers into a netlist file, where all lines from memory output to instruction register are traced, all keys are located and addresses linked to decryption stages. Total time needed would be slightly higher than LCE, but costs far less in equipement. Automation of most reverse engineering steps allow to cut time needed from months to a matter of days: the possibility to reduce nearly to zero the time needed for this attack plus the negligible equipement costs make this approach the winning

one for security evaluations.

8 Scripts

```
##################################
NbStripes=138
NbWorldLines=NbStripes*5


WLAddress=[0b0]*10
StripeSelect=[0b0]*17
Control=0b1
WordLineSelect=[0b1]*4
##################################
def invertBit(bit):
  if(bit==0b1):
    result=0b0
  else:
    result=0b1

  return result
##################################
def getActiveStripe():
  stripeIndex=-1
  lastSelector=-1
  midSelector=-1
  firstSelector=-1

  found=0
  counter=12
  while(found==0 and counter<17):
    if(StripeSelect[counter]==0b1):
      found=1
      lastSelector=counter
    else:
      counter+=1
```

```
  found=0
  counter=4
  while(found==0 and counter<12):
    if(StripeSelect[counter]==0b1):
      found=1
      midSelector=counter
    else:
      counter+=1

  found=0
  counter=0
  while(found==0 and counter<4):
    if(StripeSelect[counter]==0b1):
      found=1
      firstSelector=counter
    else:
      counter+=1

  if(Control==0b1 and firstSelector!=-1 and midSelector!=-1 and
   lastSelector!=-1):
   #print "first %d - mid %d - last %d"%(firstSelector,
   midSelector,lastSelector)
    stripeIndex=(lastSelector-12)*32
    stripeIndex+=(midSelector-4)*4
    stripeIndex+=firstSelector
   #print stripeIndex

  if(False and stripeIndex==-1):
    print "no active stripe"
  return stripeIndex
################################
def getActiveWLIndex(stripeIndex):
  WLIndex=-1
  counter=0
  found=0
  while(found==0 and counter<4):
    if(WordLineSelect[counter]==0b1):
      found=1
```

```
        #print counter
        WLIndex=(stripeIndex*4)+counter+1
      else:
        counter+=1

  if(False and WLIndex==-1):
    print "no active WL"
  return WLIndex
################################
def convertWLAddressToWLIndex(WLA, verbose):
  selectedWL=-1

  s79=WLA[3]
  s78=WLA[4]
  s76=WLA[5]

  s107=WLA[0]
  s106=WLA[1]
  s105=WLA[2]

  s104=WLA[6]
  s103=WLA[7]

  s109=WLA[8]
  s108=WLA[9]

  s1=invertBit(s104) & invertBit(s103)
  s2=invertBit(s104) & s103
  s3=s104 & invertBit(s103)
  s4=s104 & s103

  s28=invertBit(s79) & invertBit(s78) & invertBit(s76)
  s29=invertBit(s79) & invertBit(s78) & s76
  s30=invertBit(s79) & s78 & invertBit(s76)
  s31=invertBit(s79) & s78 & s76
  s32=s79 & invertBit(s78) & invertBit(s76)
  s33=s79 & invertBit(s78) & s76
  s34=s79 & s78 & invertBit(s76)
```

```
s35=s79 & s78 & s76

s37=invertBit(s107) & invertBit(s106) & invertBit(s105)
s36=invertBit(s107) & invertBit(s106) & s105
s39=invertBit(s107) & s106 & invertBit(s105)
s38=invertBit(s107) & s106 & s105
s40=s107 & invertBit(s106) & s105

s_wl0=invertBit(s109) & invertBit(s108)
s_wl1=invertBit(s109) & s108
s_wl2=s109 & invertBit(s108)
s_wl3=s109 & s108

StripeSelect[0]=s1
StripeSelect[1]=s2
StripeSelect[2]=s3
StripeSelect[3]=s4

StripeSelect[4]=s28
StripeSelect[5]=s29
StripeSelect[6]=s30
StripeSelect[7]=s31
StripeSelect[8]=s32
StripeSelect[9]=s33
StripeSelect[10]=s34
StripeSelect[11]=s35

StripeSelect[12]=s36
StripeSelect[13]=s37
StripeSelect[14]=s38
StripeSelect[15]=s39
StripeSelect[16]=s40

WordLineSelect[0]=s_wl0
WordLineSelect[1]=s_wl1
WordLineSelect[2]=s_wl2
WordLineSelect[3]=s_wl3
```

```
  selectedStripe=getActiveStripe()
  selectedWL=getActiveWLIndex(selectedStripe)

  if(verbose==1):
    printFromWLAddressToWLIndex(WLA, StripeSelect , WordLineSelect ,
    selectedStripe ,selectedWL)

  return (selectedStripe ,selectedWL)
##################################
def printFromWLAddressToWLIndex(WLA, SSel ,WLS, SS ,SWL):
  print "current condition"
  print "world line addresses"
  print WLA
  print "intermediate node"
  print "stripe selection with only one active bit for range
   [0:3], [4:11],[12:16]"
  print SSel
  print "world line selection"
  print WLS
  print "retrieving world line index"
  print "selected %dth stripe"%SS
  print "selected world line %d (pass transistors excluded"%SWL
  return
##################################
def swingTest():
  print "swing test"
  for i in range(0,1024):
    WLAdd=[0b0]*10
    binary='{0:010b}'.format(i)
    for j in range(0,10):
      if(binary[j]=='0'):
        WLAdd[j]=0b0
      else:
        WLAdd[j]=0b1
    #print WLAdd
    (SIndex ,WLindex)=convertWLAddressToWLIndex(WLAdd,0)
    if(WLindex>0):
      print "address : %d\t0b%s addresses Stripe %d and WL %d"%(i ,
```

```
    binary,SIndex,WLindex)
  return
#################################
print "script starts"

Control=0b1
print "enable "+bin(Control)

WLAddress[0]=0b0
WLAddress[1]=0b0
WLAddress[2]=0b0
WLAddress[3]=0b0
WLAddress[4]=0b0
WLAddress[5]=0b0
WLAddress[6]=0b0
WLAddress[7]=0b0
WLAddress[8]=0b0
WLAddress[9]=0b0

convertWLAddressToWLIndex(WLAddress,1)

swingTest()

print "end script"
#################################
```

Listing 1: Multiplexing logic emulation

```
#################################
NbStripes=138
NbWorldLines=NbStripes*5

StripeSelect=[0b0]*17
Control=0b1
WordLineSelect=[0b1]*4

#################################
def getActiveStripe():
  stripeIndex=-1
```

```
lastSelector=-1
midSelector=-1
firstSelector=-1

found=0
counter=12
while(found==0 and counter<17):
  if(StripeSelect[counter]==0b1):
    found=1
    lastSelector=counter
  else:
    counter+=1

found=0
counter=4
while(found==0 and counter<12):
  if(StripeSelect[counter]==0b1):
    found=1
    midSelector=counter
  else:
    counter+=1

found=0
counter=0
while(found==0 and counter<4):
  if(StripeSelect[counter]==0b1):
    found=1
    firstSelector=counter
  else:
    counter+=1

if(Control==0b1 and firstSelector!=-1 and midSelector!=-1 and
 lastSelector!=-1):
  stripeIndex=(lastSelector-12)*32
  stripeIndex+=(midSelector-4)*4
  stripeIndex+=firstSelector+1

if(stripeIndex==-1):
```

```
    print "no active stripe"
  return stripeIndex
#################################
def getActiveWLIndex(stripeIndex):
  WLIndex=-1
  counter=0
  found=0
  while(found==0 and counter<4):
    if(WordLineSelect[counter]==0b0):
      found=1
      WLIndex=(stripeIndex*4)+counter+1
    else:
      counter+=1

  if(WLIndex==-1):
    print "no active WL"
  return WLIndex
#################################
print "script starts"

Control=0b1
StripeSelect[3]=0b1
StripeSelect[5]=0b1
StripeSelect[16]=0b1
WordLineSelect[2]=0b0
print "current condition"
print Control
print StripeSelect
print WordLineSelect
print "retrieving world line index"
selectedStripe=getActiveStripe()
print selectedStripe
selectedWL=getActiveWLIndex(selectedStripe)
print selectedWL
print "end script"
#################################
```

Listing 2: Multiplexing logic emulation

```
OutputType=[0]*16
Outputs=[0]*16

GndLines=[0b0]*8
BitlineDrivers=[0b0]*8
offsetType2=5

def inv(a):
    if(a==0b1):
        b=0b0
    elif(a==0b0):
        b=0b1
    return b

def nand2(a, b):
    c=0b1
    if(a==0b0 & b==0b0):
        c=0b1
    elif(a==0b0 & b==0b1):
        c=0b1
    elif(a==0b1 & b==0b0):
        c=0b1
    elif(a==0b1 & b==0b1):
        c=0b0
    return c

def nor2(a, b):
    d=0b0
    if(a==0b0 & b==0b0):
        d=0b1
    elif(a==0b0 & b==0b1):
        d=0b0
    elif(a==0b1 & b==0b0):
        d=0b0
    elif(a==0b1 & b==0b1):
        d=0b0
    return d
```

```
def nand3(a, b, c):
    d=inv((a & b & c))
    if(d==0):
        e=0b0
    else:
        e=0b1
    return e

def getBlock():
    NColumn = 0
    for i in range(0,8):
        if(GndLines[i]==0b1):
            NColumn=i
    return NColumn

def getBitline():
    NBitline = 0
    for i in range(0,8):
        if(BitlineDrivers[i]==0b1):
            if (s12!=s76):
                if((s12==0b1) and (s76==0b0)):
                    NBitline=i+8
                else:
                    NBitline=i
            else:
                print "s12 and s76[0] should be opposite"
    return NBitline

def getBitlinefromType1():
    bitlineIndex=getBlock()*16+getBitline()
    return bitlineIndex

def getBitlinefromType2(a, b):

    s12=a
    s76=b

    bitlineIndex=-1
```

```
NColumn=getBlock()

if(NColumn<3):
    bitlineIndex=getBitlinefromType1()
elif(NColumn>4):
    bitlineIndex=getBitlinefromType1() + offsetType2
else: ##4 cases for the 2 central blocks
    if(GndLines[3]==1):
        for i in range(0,8): #calculate relative bitline index
within the block
            if(BitlineDrivers[i]==0b1):
                if (s12!=s76):
                    if((s12==0b1) and (s76==0b0)):
                        NBitline=16-i
                    else:
                        NBitline=8-i
                else:
                    print "s12 and s76 should be opposite"
                    NBitline=i
                    print("bitline %s and %s are unreachable"
%(NBitline, NBitline+8))
        bitlineIndex=NColumn*16+NBitline
    elif(GndLines[3]==0 and GndLines[4]==1):
        for i in range(0,8): #calculate relative bitline index
within the block
            if(BitlineDrivers[i]==0b1):
                if (s12!=s76):
                    if((s12==0b1) and (s76==0b0)):
                        NBitline=16-i
                    else:
                        NBitline=8-i
                else:
                    print "s12 and s76 should be opposite"
                    NBitline=i
                    print("bitline %s and %s are unreachable"
%(NBitline, NBitline+8))
        bitlineIndex=NColumn*16+NBitline
return bitlineIndex
```

```
print "script starts"

GndLines=[0b0]*8
BitlineDrivers=[0b0]*8

OutputType[0]=1
OutputType[1]=2
OutputType[2]=1
OutputType[3]=2
OutputType[4]=1
OutputType[5]=2
OutputType[6]=1
OutputType[7]=2
OutputType[8]=1
OutputType[9]=2
OutputType[10]=1
OutputType[11]=2
OutputType[12]=1
OutputType[13]=2
OutputType[14]=1
OutputType[15]=2


for k in range(0,8):
    GndLines[k]=0b1
    print ("GndLines: %s" % GndLines)
    print ("sending out block: %s" %k)
    for j in range(0,8):
        BitlineDrivers[j]=0b1
        print ("BitlineDrivers: %s" % BitlineDrivers)
        s12=0b0 #s185
        print("s12: %s" %s12)
        s76=0b1 #s181
        print("s76: %s" %s76)
        typ2_counter = 0
        for i in range(0,16):
            offset=128*i+5*typ2_counter
```

```
            if (OutputType[i]==1):
                Outputs[i]=getBitlinefromType1()+offset
            else:
                Outputs[i]=getBitlinefromType2(s12, s76)+offset
                typ2_counter=typ2_counter + 1
        for i in range(0,16):
            print ("Outputs[%s]: %s"%(i, Outputs[i]))
        s12=0b1 #s185
        print("s12: %s" %s12)
        s76=0b0 #s181
        print("s76: %s" %s76)
        typ2_counter = 0
        for i in range(0,16):
            offset=128*i+5*typ2_counter
            if (OutputType[i]==1):
                Outputs[i]=getBitlinefromType1()+offset
            else:
                Outputs[i]=getBitlinefromType2(s12, s76)+offset
                typ2_counter=typ2_counter + 1
        for i in range(0,16):
            print ("Outputs[%s]: %s"%(i, Outputs[i]))
        BitlineDrivers[j]=0b0
    GndLines[k]=0b0
print "end script"
```

Listing 3: Multiplexing logic emulation

References

[1] ARM Security Technology, *Building a Secure System using TrustZone Technology*, Copyright 2005-2009 ARM Limited.

[2] Ross Anderson, *Security Engineering, A Guide to Building Dependable Distributed Systems*, Chapter 16, Wiley Publishing Inc.

[3] Neil Chenoweth, *Murdoch's Pirates, Before the Phone Hacking, there was Rupert's pay-TV Skulldruggery*, Allen & Unwin

[4] Sergei P. Skorobogatov, *Semi-invasive attacks A new approach to hardware security analysis*, University of Cambridge Technical Report Number 630, UCAM-CL-TR-630 ISSN 1476-2986, April 2005

[5] Sergei P. Skorobogatov, *Hardware Security of Semiconductor Chips: Progress and Lessons* School of Computing Science, Newcastle upon Tyne, 27 June 2011

[6] Sergei P. Skorobogatov, *Security, Reliability and Backdoors*, Security Group, Computer Laboratory, University of Cambridge, 13 May 2014

[7] Chris Gerlinsky, *How Do I Crack Satellite and Cable Pay TV?*

[8] Jan M. Rabaey, Anantha Chandrakasan, Bora Nikolic, *Digital Integrated Circuits*, Second Edition, Prentice-Hall, 2002

[9] Andrew 'bunny' Huang, *Hacking the Xbox, An Introduction to Reverse Engineering*, No Starch Press, Inc, 2003

[10] Olivier Thomas, *ADVANCED IC REVERSE ENGINEERING TECHNIQUES: IN DEPTH ANALYSIS OF A MODERN SMART CARD*, Black-Hat USA 2015

[11] Gedare Bloom, Eugen Leontie, Bhagirath Narahari, Rahul Simha *Hardware and Security: Vulnerabilities and Solutions*, Handbook on Securing Cyber-Physical Critical Infrastructure, chapter 12, 2012, Elsevier Inc.

[12] Wolfgang Rankl and Wolfgang Effing, *Smart Card Handbook*, Third Edition, John Wiley & Sons Ltd, 2003

[13] Christofer Tarnovsky, *Security Failures In Secure Devices*, BlackHat DC, February 21, 2008,

[14] Nick Chen, *The Benefits of Antifuse OTP*, Semiconductor Engineering, December 19, 2016 https://semiengineering.com/the-benefits-of-antifuse-otp/

[15] Rich Smith, *PhlashDance:Discovering permanent denial of service attacks against embedded systems*, HP Labs, EUSecWest 08, May 22, 2008

[16] Kelly Jackson Higgins, *Permanent Denial-of-Service Attack Sabotages Hardware*, DarkReading, May 19, 2008 https://www.darkreading.com/permanent-denial-of-service-attack-sabotages-hardware/d/d-id/1129499

[17] Abhinav Singh, *The Underground Ecosystem of Credit Card Frauds*, BlackHat Asia, March, 2015 https://www.blackhat.com/docs/asia-15/materials/asia-15-Singh-The-Underground-Ecosystem-Of-Credit-Card-Frauds-wp.pdf

[18] Cibersecurity unit of Kudelski Group *The New Face of PayTV Piracy and How to Fight It*, 2016

[19] https://en.wikipedia.org/wiki/Dreambox

[20] H. Cruickshank, M.P. Howarth, S.Iyengar, Z. Sun1 *A Comparison between satellite DVB conditional access and secure IP multicast*

[21] Oliver Kommerling, Markus G. Kuhn, *Design Principles for Tamper-Resistant Smartcard Processors*, USENIX Workshop on Smartcard Technology Chicago, Illinois, USA, May 1011, 1999

[22] Wieland Fischer, *Aspects of the Development of Secure and Fault-Resistant Hardware*, Infineon Technologies AG

[23] *Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components*, April 2017

[24] Olivier Thomas, Dmitry Nedospasov, *On the Impact of Automating the IC Analysis Process*, Texplained SARL, Technische Universitat Berlin

[25] Shahed E. Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, Mark Tehranipoor, *A Survey on Chip to System Reverse Engineering*, University of Connecticut, ACM Journal on Emerging Technologies in Computing Systems, Vol. V, No. N, Article acmArticle, Pub. date: Month 2015.

[26] Randy Torrance, Dick James, *Reverse Engineering in the Semiconductor Industry*, Chipworks Inc., 2007

[27] Clemens Helfmeier, Christian Boit, Dmitry Nedospasov, Jean-Pierre Seifert, *Cloning Physically Unclonable Functions*

[28] Tony Moor, Eli Malyanker, Efrat Raz-Moyal *Single Die 'Hands-Free' Layer-by-Layer Mechanical Deprocessing for Failure Analysis or Reverse Engineering*, Presented at ISTFA November 2008

[29] Friedrich Beck, *Intengrated Circuit Failure Analysis: A Guide to Preparation Tecniques*, John Wiley & Sons

[30] Dmitry Nedospasov, *SECURITY OF THE IC BACKSIDE*, Technishe Universitat Berlin, 2014

[31] Andrew Zonenberg, *CSCI 4974/ 6974 Hardware Reverse Engineering, Lecture3: Depackaging*

[32] Weilie Zhou, Robert P. Apkarian, Zhong Lin Wang, and David Joy, *Fundamentals of Scanning Electron Microscopy*

[33] JEOL Ltd., *Scanning Electron Microscope A To Z, Basic Knowledge For Using The SEM*

[34] Prashanth Prasad, *Preparation of a cross-sectional semiconductor IC device sample for SEM observation*, Gatan

[35] Pamela Samuelson, Suzanne Scotchmer *THE LAW & ECONOMICS OF REVERSE ENGINEERING*

[36] DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL

[37] Apostolos P. Fournaris, Lidia Pocero Fraile and Odysseas Koufopavlou, *Exploiting Hardware Vulnerabilities to Attack Embedded System Devices: a Survey of Potent Microarchitectural Attacks*, MDPI, Electronics, 2017

[38] WIPO *Patent Expert Issues: Layout Designs (Topographies) of Integrated Circuits*, *http* : *//www.wipo.int/patents/en/topics/integrated*$_c$*ircuits.html*