

UNIT 5

HARDWARE SECURITY

Introduction to hardware security, Hardware Trojans, Side – Channel Attacks – Physical Attacks and Countermeasures – Design for Security. Introduction to Blockchain Technology

INTRODUCTION TO HARDWARE SECURITY

- Emerging trends in electronic hardware production, such as intellectual property-based (IP-based) system on chip (SoC) design, and a long and distributed supply chain for manufacturing and distribution of electronic components leading to reduced control of a chip manufacturer on the design and fabrication steps—have given rise to many growing security concerns.
- This includes malicious modifications of ICs, also referred to as Hardware Trojan attacks, in an untrusted design house or foundry.
- Other examples include side-channel attacks, where secret information of a chip can be extracted through measurement and analysis of side-channels, that is, physical signals, such as power, signal propagation delay, and electromagnetic emission.

HARDWARE SECURITY VS. HARDWARE TRUST

- Hardware security issues arise from its own vulnerability to attacks (e.g., side-channel or Trojan attacks) at different levels (such as, chip or PCB), as well as from lack of robust hardware support for software and system security.
- On the other hand, hardware trust issues arise from involvement of untrusted entities in the life cycle of a hardware, including untrusted IP or computer-aided design (CAD) tool vendors, and untrusted design, fabrication, test, or distribution facilities. These parties are capable of violating the trustworthiness of a hardware component or system. They can potentially cause deviations from intended functional behaviour, performance, or reliability.

ATTACKS, VULNERABILITIES, AND COUNTERMEASURES

ATTACK VECTORS

Attack vectors as they relate to hardware security are means or paths for attackers to get access to hardware components for malicious purposes, for example, to compromise it or extract secret assets stored in hardware. Example of hardware attack vectors are side-channel attacks, Trojan attacks, IP piracy, and PCB tampering.

ATTACK SURFACE

Attack surface is the sum of all possible security risk exposures. It can also be explained as the aggregate of all known, unknown, and potential vulnerabilities, and controls across all hardware, software, and network components.



Fig. Possible attack surfaces of a Smartphone

Figure illustrates major attack surfaces of a smartphone, composed of software, network, data, and hardware components. Keeping the attack surface as small as possible is a common goal for developing counter measures. With respect to hardware security, three main attack surfaces are as follows.

Chip Level Attacks:

Chips can be targeted for reverse engineering, cloning, malicious insertion, side-channel attacks, and piracy. Counterfeit or fake chips can be sold as original units if the attacker can create a copy that has a similar appearance or features as the original. Trojan-infected chips can also find their place in the supply chain, which can pose a threat of unauthorized access, or malfunction. Side-channel attacks can be mounted on a chip with the goal to extract secret information stored inside it.

PCB-Level Attacks:

PCBs are common targets for attackers, as they are much easier to reverse engineer and tamper than ICs. Design information of most modern PCBs can be extracted through relatively simple optical inspection (for example, X-Ray tomography) and efficient signal processing. Primary goals for these attacks are to reverse engineer the PCB, and obtain the schematic of the board to redesign it and create fake units.

System-Level Attacks:

Complex attacks involving the interaction of hardware-software components can be mounted on the system. By directly focusing on the most vulnerable parts in a system, such as DFT infrastructure at PCB level (for example, JTAG) and memory modules, attackers may be able to compromise the system's security by gaining unauthorized control and access to sensitive data.

VULNERABILITIES

- Vulnerabilities refer to weakness in hardware architecture, implementation, or design/test process, which can be exploited by an attacker to mount an attack.
- These weaknesses can either be functional or nonfunctional, and they vary based on the nature of a system and its usage scenarios.

Following is a description of some typical vulnerability in hardware systems:

Functional Bug:

- Most vulnerability is caused by functional bugs and poor design/testing practices.
- They include weak cryptographic hardware implementation and inadequate protection of assets in an SOC.
- Attackers may find these vulnerabilities by analyzing the functionality of a system for different input conditions to look for any abnormal behaviors.
- Additionally, vulnerabilities may be discovered accidentally, which makes it easier for an attacker to perform malicious activities using these newly discovered issues in the system.

Side-Channel Bug:

- These bugs represent implementation level issues that leak critical information stored inside a hardware component (for example processors or crypto chips) through different forms of side-channels.
- Attackers may find these vulnerabilities by analyzing the side-channel signals during operation of a hardware component.

Test/Debug infrastructure:

- Most hardware systems provide a reasonable level of testability and debuggability, which enable designers and test engineers to verify the correctness of operation.
- They also provide means to study internal operations and processes running in hardware, which are essential for debugging hardware.
- These infrastructures, however, can be misused by attackers, where extraction of sensitive information or unwanted control of a system can be possible using the test/debug features.

Access control or information-flow issues:

- In some cases, a system may not distinguish between authorized and unauthorized users. This vulnerability may give an attacker access to secret assets and functionality that can be misused or leveraged.
- Moreover, an intelligent adversary can monitor the information flow during system operation to decipher security critical information, such as, control flow of a program and memory address of a protected region from hardware.

COUNTERMEASURES

Countermeasures can either be employed at design or test time.

Design solutions:

- Design-for-security (DfS) practices have emerged as powerful countermeasures.
- DfS offers effective low-overhead design solutions that can provide active or passive defense against various attacks. DfS techniques, such as obfuscation, use of reliable security primitives, side channel resistance (for example, masking and hiding techniques), and hardening schemes for Trojan insertion, can reliably protect against many major attack vectors.

Test and verification solutions:

- Test and verification techniques have constituted a major category of protection approaches against the diverse security and trust issues.
- Both pre-silicon verification functional as well as formal and post-silicon manufacturing testing have been considered as mechanisms to identify security vulnerabilities and trust issues for chips, PCBs, and systems.

HARDWARE TROJANS

- A hardware Trojan (HT) is defined as a malicious, intentional modification of a circuit design that results in undesired behavior when the circuit is deployed.
- SoCs that are 'infected' by a hardware Trojan may experience changes in their functionality or specification, may leak sensitive information, or may experience degraded or unreliable performance.
- Hardware Trojan poses a serious threat to any hardware design being deployed in a critical operation.
- As the hardware Trojans are inserted at the hardware level, software-level countermeasures may be inadequate to address the threat posed by HT.
- Also, detection of Trojans in a hardware design is challenging as there is no golden version against which to compare a given design during verification.
- In theory, an effective way to detect a Trojan is to activate the Trojan and observe its effects, but a Trojan's type, size, and location are unknown, and its activation is, most likely, a rare event.

HARDWARE TROJAN STRUCTURE

- The basic structure of a Trojan includes two main parts, trigger and payload.
- A Trojan trigger decides when the Hardware Trojan or HT will wake up.
- Payload decides what will happen when the Trojan will wake up.
- Once the trigger detects an expected event or condition, the payload is activated to perform malicious behavior.
- Typically, the trigger is expected to be activated under extremely rare conditions, so the payload remains inactive most of the time. When the payload is inactive, the IC acts like a Trojan-free circuit, making it difficult to detect the Trojan.

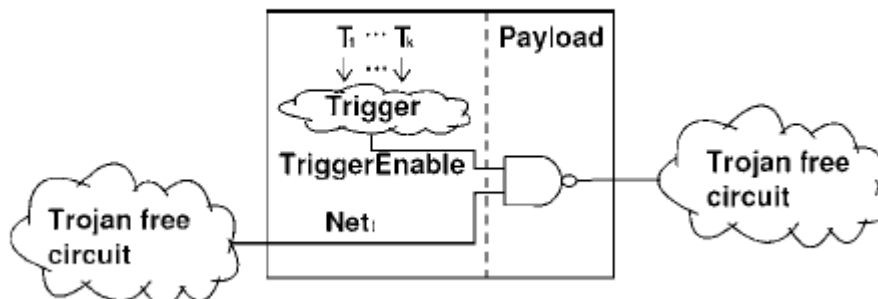


Figure shows the basic structure of the Trojan at gate level. The trigger inputs (T_1, T_2, \dots, T_k) come from various nets in the circuit. The payload taps the original signal Net_i from the original (Trojan-free) circuit and the output of the Trigger. Since the trigger is expected to be activated under rare condition, the payload output stays at the same

value most of the time, Net_i . However, when the trigger is active, that is, Trigger Enable is "0", the payload output will be different from Net_i ; this could result in injecting an erroneous value into the circuit and causing error at the output.

TROJAN MODELING

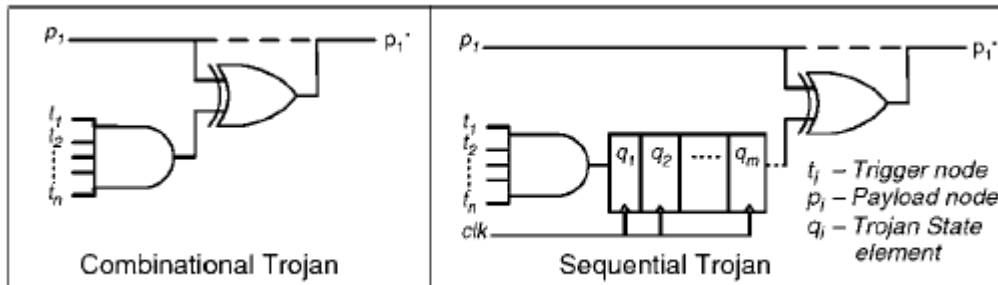


Figure shows generic models for combinational and sequential Trojans.

The trigger condition is an n -bit value at internal nodes, which is assumed to be rare enough to evade normal functional testing. The payload is defined as a node that is inverted when the Trojan is activated.

To make it more difficult to detect, one might consider a sequential Trojan, which requires the rare event to repeat 2^m times before the Trojan gets activated and inverts the payload node.

HARDWARE TROJAN EXAMPLES

Trojans in Cryptographic Engines

A possible Trojan attack in a crypto engine can try to subvert the security mechanisms. The payload provides a mechanism that presents dummy keys, predefined by the attacker, instead of the actual cryptographic keys used for sensitive encryption or signature verification operations, to leaking the secret hardware keys via covert side-channels.

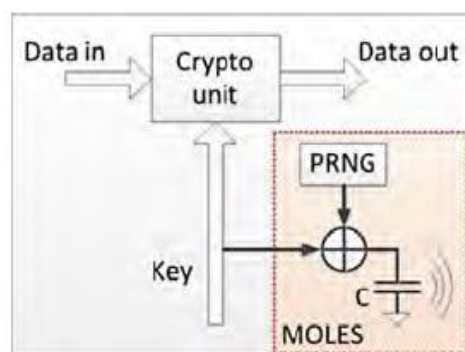


Figure provides an example of such a Trojan, which attempts to leak a secret key from inside a cryptographic module through power side-channels using a technique called malicious off-chip leakage, enabled by side-channels (MOLES).

Trojans in General-Purpose Processors

- Modern processors implement a hardware chain of trust to ensure that malware cannot compromise the hardware assets, such as secret keys and memory range protections.
- However, in such systems, the attacker at an untrusted fabrication facility could implement a backdoor, which disables the secure booting mechanism under certain rare conditions or when presented with a unique rare input condition in the hands of an end-user adversary.

HARDWARE TROJANS IN FPGA DESIGNS

- FPGAs are widely used today in an array of embedded applications, ranging from telecommunications and data centers to missile guidance systems.
- FPGA based Trojans can be in the form of IP blocks, which get loaded onto a generic FPGA fabric and cause malicious activity on the system in which the FPGA is deployed.
- FPGAs contain a large volume of reconfigurable logic in the form of lookup tables, block RAM, and programmable interconnects, which can be used to realize any arbitrary sequential or combinational design.
- However, there might be a significant amount of reconfigurable logic open to a malicious party who can load a hardware Trojan and affect the FPGA integrated system or compromise the IP loaded onto the FPGA.

These FPGA device specific hardware Trojans and their effects are explained in and summarized below.

1. ACTIVATION CHARACTERISTIC

FPGA device based hardware Trojans can either be IP-dependent or IP-independent.

IP-Dependent Trojans

A malicious foundry or FPGA vendor may implement a hardware Trojan that can monitor the logic values of several lookup tables (LUTs) in the FPGA fabric.

Once triggered, such Trojans can corrupt other LUT values, load incorrect values into block RAMs (BRAMs), or sabotage configuration cells.

IP-Independent Trojans

A malicious foundry or vendor may also implement a Trojan into an FPGA chip that is completely independent of the IP loaded onto it. Such Trojans can occupy a small portion of FPGA resources and malfunction IP-independent but critical FPGA resources, such as digital clock managers (DCM).

2. PAYLOAD CHARACTERISTICS

FPGA device based Trojans can also bring about unique malicious effects, such as causing malfunction of FPGA resources or leakage of the IP loaded onto the FPGA.

Malfunction

- Hardware Trojans in FPGA devices can either cause logical malfunction by corrupting LUT(s) or SRAM values, thereby affecting the functionality of the implemented IP, or by causing physical damage to the FPGA device.
- For example, a triggered hardware Trojan could reprogram an I/O port set as an input to become an output I/O port, while suppressing the configuration cells that prevent it from being programmed as such. This would cause a high short-circuit current to flow between the FPGA and the system it is connected to, thereby leading to physical device failure.

IP Leakage

- FPGAs today offer bit stream encryption capabilities in order to protect the IP loaded onto an FPGA device. However, such encryption only prevents a direct or unauthorized readback by software.
- A hardware Trojan may circumvent such protection by either leaking the decryption key, or even the entire IP. The Trojan may tap the decryption key as it comes out of non-volatile memory, or the actual decrypted IP, which could then be exfiltrated either via covert side-channels.

HARDWARE TROJANS TAXONOMY

Taxonomy is developed based on hardware Trojans' physical, activation, and functional characteristics. In this regard, hardware Trojans are classified based on five attributes: (1) insertion phase, (2) abstraction level, (3) activation mechanism, (4) payload, and (5) location.

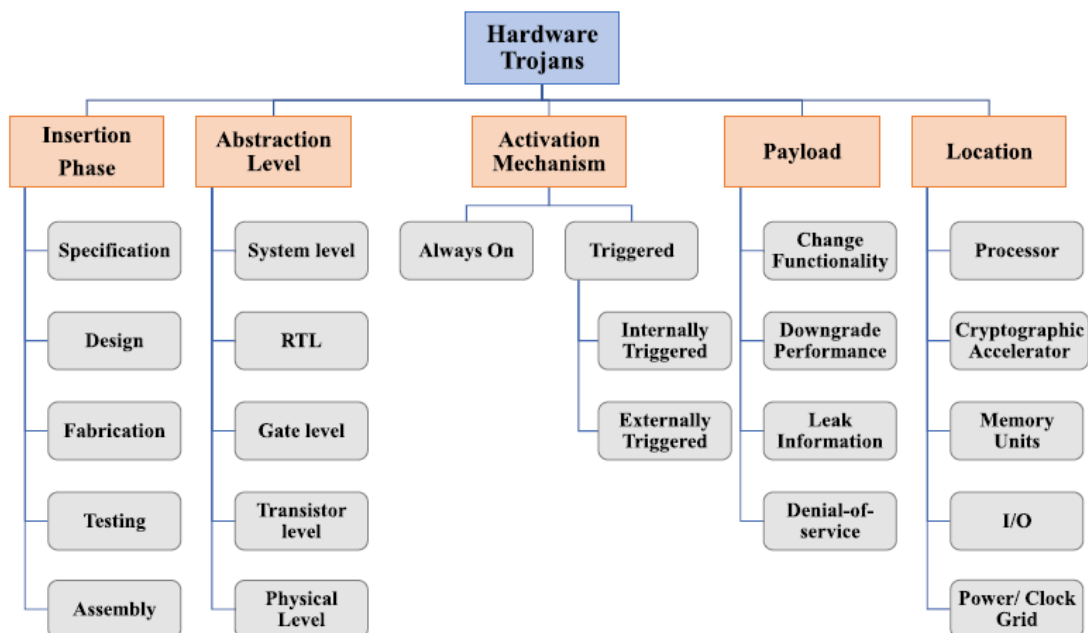


Fig : Taxonomy of hardware Trojans

1. INSERTION PHASE

Trojans can also be classified based on the phases in which they are inserted.

a. Specification Phase

In this phase, chip designers define the system's characteristics: the target environment, expected function, size, power, and delay. Functional specifications or other design constraints can be altered. For example, a Trojan at the specification phase might change the hardware's timing requirements.

b. Design Phase

Developers consider functional, logical, timing and physical constraints as they map the design on to the target technology. At this point, they can use third party IP blocks and

standard cells. Trojans might be in any of the components that aid the design. For example, a standard cell library can be tampered with Trojans.

c. Fabrication Phase

- During this phase, developers create a mask set and use wafers to produce the masks. Subtle mask changes can have serious effects.
- In an extreme case, an adversary can substitute a different mask set. Chemical compositions might be altered during fabrication to increase the electro migration in critical circuitry, such as power supplies and clock grids, which would accelerate failures.

d. Testing Phase

- The IC testing phase is important for hardware trust, not only because it is a likely phase for Trojan insertion, but also because it provides an opportunity for Trojan detection. Testing is only useful for detection when done in a trustworthy manner.
- For example, an adversary who inserted a Trojan in the fabrication phase would want to have control over the test vectors to ensure that the Trojan is not detected during test.

e. Assembly Phase

- Developers assemble the tested chip and other hardware components on a printed circuit board (PCB).
- Every interface in a system, where two or more components interact is a potential Trojan insertion site.
- Even if all the ICs in a system are trustworthy, malicious assembly can introduce security flaws in the system. For example, an unshielded wire connected to a node on the PCB can introduce unintended electromagnetic coupling between the signal on the board and its electromagnetic surroundings. An adversary can exploit this for information leakage and fault injection.

2. ABSTRACTION LEVEL

Trojan circuits can be inserted at various hardware abstraction levels.

a. System Level

At the system level, the different hardware modules, interconnections, and communication protocols used are defined. At this level, the Trojans may be triggered by the modules in the target hardware. For example, the ASCII values of the inputs from the keyboard can be interchanged.

b. Register-Transfer Level

At the RTL, chip designers describe each functional module in terms of registers, signals, and Boolean functions. A Trojan can be easily designed and inserted at the RTL because the attacker has full control over the design's functionality.

c. Gate Level

At this level, a SoC is represented as an interconnection of logic gates. An attacker can carefully control all aspects of the inserted Trojan, including its size and location. For example, a Trojan might be a simple comparator consisting of basic gates (AND, OR, XOR gates) that monitor the chip's internal signals.

d. Transistor Level

Chip designers use transistors to build logic gates. This level gives the Trojan designer control over circuit characteristics, such as power and timing. The attacker can insert or remove individual transistors, altering the circuit functionality, or modify transistor

sizes to alter circuit parameters. For example, a transistor-level Trojan might be a transistor with low gate width that can cause more delay in the critical path.

e. Physical Level

An attacker can insert Trojans by modifying the size of the wires and distances between circuit elements and reassigning metal layers. For example, changing the width of the clock wires, timing critical nets or metal wires in the chip can cause clock skew.

3. ACTIVATION MECHANISM

Some Trojans are designed to be always on, whereas others remain dormant until triggered. A triggered Trojan needs an internal or external event to be activated. Once the trigger activates a Trojan, it can remain active forever or return to a dormant state after a specified time.

a. Internally Triggered

An internally triggered Trojan is activated by an event within the target device. The event might be either time-based or physical condition based. A counter in the design can trigger a Trojan at a predetermined time, resulting in a silicon timebomb. Similarly, a Trojan can trigger when the chip temperature exceeds a certain threshold.

b. Externally Triggered

- An externally triggered Trojan requires external input to the target module to activate. The external trigger can be user input or component output.
- User input triggers include push buttons, switches, keyboards, or keywords and phrases in the input data stream in a system.
- Component output triggers might be from any of the components that interact with the target device.
- For example, a Trojan in a crypto module can derive its trigger condition from the applied plaintext input and triggers when it observes a specific plaintext, or a combination of plaintext and operating conditions.

One can also classify the triggered Trojans into two categories:

- (1) Analog triggered and
- (2) Digital triggered.

Analog-triggered Trojans are triggered analog signals, such as temperature and voltage. Digitally triggered Trojans are triggered by logic-conditions, such as state of the flip flops, state of a logic net, counter, clock signal, data, instruction, and/or interrupts.

4. PAYLOAD

Trojans can also be characterized by their payload, that is, the malicious effects caused by them when they become activated.

a. Change Functionality

Trojan can change the functionality of the target device, and cause subtle errors that might be difficult to detect during manufacturing test. For example, a Trojan might cause an error detection module to accept inputs that should be rejected.

b. Downgrade Performance

A Trojan can downgrade performance by intentionally changing device parameters. These include functional, interface, or parametric characteristics, such as power and delay. For example, a Trojan might insert more buffers in the chip's interconnections and, hence, consume more power, which in turn could drain the battery quickly.

c. Leak Information

A Trojan can also leak information through both covert and overt channels. Sensitive data can be leaked via radio frequency, optical or thermal power, timing side channels,

and interfaces, such as RS-232 and JTAG (Joint Test Action Group). For example, a Trojan might leak a cryptographic algorithm's secret key through unused RS-232 ports.

d. Denial-of-Service

A denial-of-service (DoS) Trojan can cause the target module to exhaust scarce resources, such as bandwidth, computation, and battery power. It could also physically destroy, disable, or alter the device's configuration, for example, causing the processor to ignore the interrupt from a specific peripheral. DoS can be either temporary or permanent.

5. LOCATION

A hardware Trojan can be inserted in a single component or spread across multiple components, for example, processor, memory, input/output, power supply, or clock grid. Trojans distributed across multiple components can act independently of one another or together as a group to accomplish their attack objectives.

a. Random Logic

A Trojan can be inserted into the random logic portion of an SoC. Detection of such Trojans are extremely challenging since understanding functionality of random logic is difficult, hence making it limited to generate effective test stimuli. Size of such logic in a SoC can be quite large.

b. Processing Unit

Any Trojan embedded into the logic units that are part of the processor can be grouped under this category. A Trojan in the processor might, for example, change the instructions' execution order.

c. Cryptographic Accelerator

A Trojan in a cryptomodule can leak the secret key (attack on confidentiality) or replace the key (attack on integrity), and compromise the security of the overall system.

d. Memory Units

Trojans in the memory blocks and their interface units fall in this category. These Trojans might alter the value stored in the memory and also block read or block write access to certain memory locations; for example, change the contents of a programmable read-only memory in a SoC.

e. Input/Output Port

Trojans can reside in a chip's peripherals or within the PCB. These peripherals interface with the external components and can give the Trojan control over data communication between the processor and the system's external components. For example, a Trojan might alter the data coming through a JTAG port.

f. Power Supply

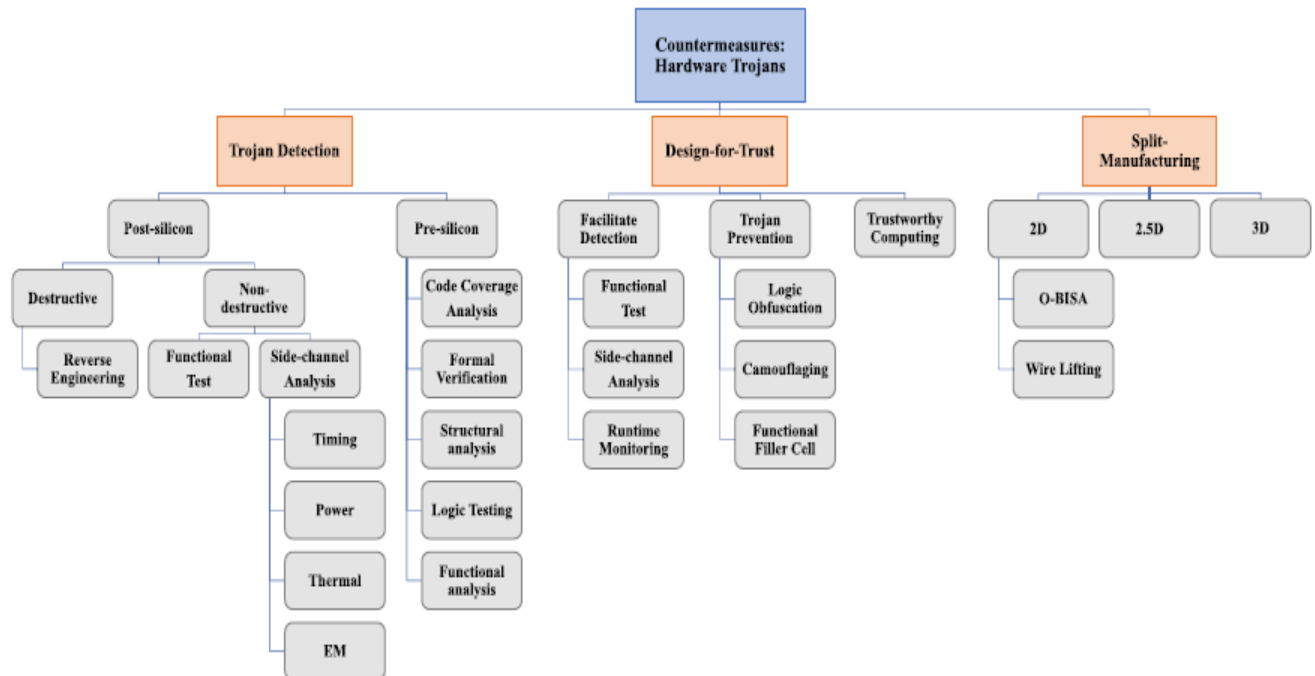
Modern SoCs include many voltage islands, a large number of locally distributed voltage regulators, and dynamic voltage/frequency systems, where the chip frequency is adjusted in the field by changing VDD as the chip ages in the field. Trojans could be inserted by an adversary to alter the voltage and current supplied to the chip, causing failures.

g. Clock Grid

Trojans in the clock grid can change the clock's frequency; insert glitches in the clock supplied to the chip, and launch fault attacks. These Trojans can also freeze the clock signal supplied to the rest of the chip's functional modules. For example, a Trojan might increase the clock signal skew supplied to specific parts of a chip, causing hold-time violation on short paths.

COUNTERMEASURES AGAINST HARDWARE TROJANS

These approaches are classified into two broad categories, Trojan Detection and Trojan Prevention, each of which can further be classified into several subcategories, as shown in Fig.



1. TROJAN DETECTION

Trojan detection is the most straightforward and commonly used approach to deal with hardware Trojans. It aims to verify the existing designs and fabricated SoCs without any supplementary circuitry. They are performed either at the design stage (that is, pre-silicon) to validate SoC designs or after the manufacturing stage (that is, post-silicon) to verify fabricated SoCs.

i) Post-silicon Trojan Detection

These techniques are employed after the chip is fabricated; they can be classified into destructive and non-destructive methods.

a) Destructive methods:

- These techniques typically use destructive reverse-engineering to de-package an IC, and obtain images of each layer in order to reconstruct the design-for-trust validation of the end product.
- Destructive reverse engineering has the potential of giving very high assurance that any malicious modification in the IC can be detected, but it comes with high cost and could take several weeks and months for an IC of reasonable complexity.
- Reverse engineering of modern complex SoCs is a tedious and error-prone process. Hence, in order to obtain the entire chip structure reverse engineered, one may use tens of ICs as de-processing and de-packaging could cause unintentional errors in the reverse engineering process. Therefore, in general, destructive approaches do not seem viable for Trojan detection.

b) Non-destructive methods:

Functional tests:

- These techniques attempt to activate Trojans by applying test vectors and comparing the responses with the correct results. To be effective, such techniques require availability of a golden response.
- Intelligent adversaries can design Trojans that are activated under very rare conditions, so they can go undetected under structural and functional tests during the manufacturing test process.

Side-channel signal analysis:

- These approaches detect hardware Trojans by measuring circuit parameters, such as delay, power (transient and leakage power), temperature, and radiation.
- The majority of the detection techniques assume that “golden ICs” (Trojan-free ICs) are available for comparison in order to identify Trojan-infected ICs.
- Side-channel analysis methods may succeed in detecting Trojans to some degree. However, their difficulty lies in achieving high coverage of every gate or net, and in extracting the tiny, abnormal side-channel signals of hardware Trojans in the presence of process and environmental variations.
- As the feature size of ICs shrinks and the number of transistors continue to increase, the increasing levels of process variations can easily mask the small side-channel signals induced by low-overhead and rarely triggered Trojans.

ii) Pre-silicon Trojan detection

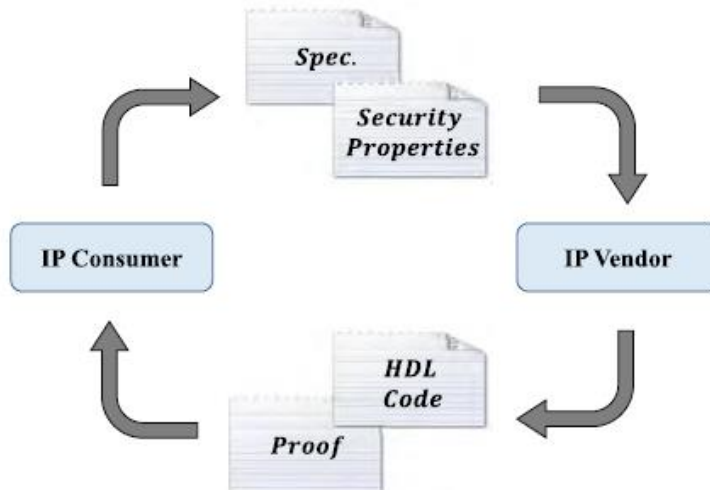
These techniques are used to help SoC developers and design engineers to validate third-party IP (3PIP) cores and their final designs. Existing pre-silicon detection techniques can be broadly classified into code coverage analysis, formal verification, structural analysis, logic testing, and functional analysis.

A) Code Coverage Analysis:

Code coverage is defined as the percentage of lines of code that has been executed during functional verification of the design. This metric gives a quantitative measure of the completeness of the functional simulation of the design. Code coverage analysis can also be applied to identify suspicious signals that may be a part of a Trojan, and validate the trustworthiness of a 3PIP.

B) Formal Verification:

- Formal methods, such as symbolic execution, model checking, and information flow have been traditionally applied to software systems for finding security bugs and improving test coverage. Formal verification has also been shown to be effective in verifying the trustworthiness of 3 PIP .
- These approaches are based on the concept of proof-carrying code (PCC) to formally validate the security-related properties of an IP.
- In these approaches, a SoC integrator provides a set of security properties in addition to the standard functional specification to the IP vendor. A formal proof of these properties, alongside the hardware IP, is then provided by the third-party vendor.
- SoC integrator then validates the proof by using the PCC. Any malicious modification of the IP would violate this proof, thereby indicating the presence of hardware Trojan.



C) Structural Analysis:

- Structural analysis employs quantitative metrics to mark signals or gates with low activation probability as suspicious.
- A metric named 'Statement Hardness' is used to evaluate the difficulty of executing a statement in the RTL code. Areas in a circuit with large value of 'Statement Hardness' are more vulnerable to Trojan insertion.
- At gate level, an attacker would most likely target hard-to-detect areas of the gate level netlist to insert Trojan.
- Inserting a Trojan in hard-to-detect areas would reduce the probability to trigger the Trojan and, thereby, reduce the probability of being detected during verification and validation testing.
- The limitations of code/structural analysis techniques are that they do not guarantee Trojan detection, and manual post processing is required to analyse suspicious signals or gates, and determine if they are a part of a Trojan.

D) Logic Testing:

The principal idea of logic testing is the same as the functional tests described earlier in the post-silicon Trojan detection section. The logic testing is conducted with simulation, whereas functional tests have to be performed on a tester for applying input patterns and collecting output responses. Therefore, existing techniques for functional tests are also applicable to logic testing.

E) Functional Analysis:

Functional analysis applies random input patterns and performs functional simulation of the IP to find suspicious regions of the IP that have similar characteristics of a hardware Trojan. The basic difference between functional analysis and logic testing is that logic testing aims to apply specific patterns to activate a Trojan, whereas functional analysis applies random patterns, and these patterns are not directed to trigger the Trojan.

iii) Design-For-Trust:

These methodologies are classified into four classes according to their objectives.

A) Facilitate Detection

a) Functional Test:

- Triggering a Trojan from inputs and observing the Trojan effect from outputs are difficult due to the stealthy nature of Trojans. A large number of low controllable and low-observable nets in a design significantly hinder the possibility of activating a Trojan.
- Two outputs of a DFF, Q and \bar{Q} , is multiplexed through a 2-to-1 multiplexer, and select either of them. This extends the state space of the design and increases the possibility of exciting/propagating the Trojan effects to circuit outputs, making them detectable.

b) Side-channel Signal Analysis:

- A number of design methods have been developed to increase the sensitivity of side-channel-based detection approaches. The amount of current a Trojan can draw could be so small that it can be submerged into an envelope of noise and process variations effects; therefore, it may be undetectable by conventional measurement equipment. However, Trojan-detection capability can be greatly enhanced by measuring currents locally, and from multiple power ports/pads.
- Additionally, some newly developed structures or sensors are implemented in the circuit to provide higher detection sensitivity compared to conventional measurements. Ring oscillator (RO) structures, shadow registers, and delay elements on a set of selected short paths are inserted for path delay measurements.

c) Runtime Monitoring:

As triggering all types and sizes of Trojans during pre-silicon and post-silicon tests is very difficult, runtime monitoring of critical computations can significantly increase the level of trust with respect to hardware Trojan attacks. These runtime monitoring approaches can utilize existing or supplemental on-chip structures to monitor chips behaviours or operating conditions such as transient power and temperature.

B) Prevent Trojan Insertion

These techniques consist of preventive mechanisms that attempt to thwart hardware Trojan insertion by attackers. To insert targeted Trojans, typically attackers need to understand the function of the design first. Attackers who are not in the design house usually identify circuit functionality by reverse engineering.

Logic Obfuscation:

- Logic obfuscation attempts to hide the genuine functionality and implementation of a design by inserting built-in locking mechanisms into the original design.
- For combinational logic obfuscation, XOR/XNOR gates could be introduced at certain locations in a design. In sequential logic obfuscation, additional states are introduced in a finite state machine to conceal its functional states.

Camouflaging:

- Camouflaging is a layout-level obfuscation technique to create indistinguishable layouts for different gates by adding dummy contacts, and faking connections between the layers within a camouflaged logic gate.
- The camouflaging technique can hinder attackers from extracting a correct gate-level netlist of a circuit from the layout through imaging

different layers; in that way, the original design is protected from insertion of targeted Trojans.

Functional Filler Cell:

- Since layout design tools are typically conservative in placement, they cannot fill 100% of the area with regular standard cells in a design. The unused spaces are usually filled with filler cells or decap cells that do not have any functionality.
- Thus, the most covert way for attackers to insert Trojans in a circuit layout is replacing filler cells, and to some degree decaps, because removing these nonfunctional cells has the smallest impact on electrical parameters.
- The built-in self-authentication (BISA) approach fills all white spaces with functional filler cells during layout design. The inserted cells are then connected automatically to form a combinational circuitry that could be tested. A failure during later testing denotes that functional filler has been replaced by a Trojan.

C) Trustworthy Computing

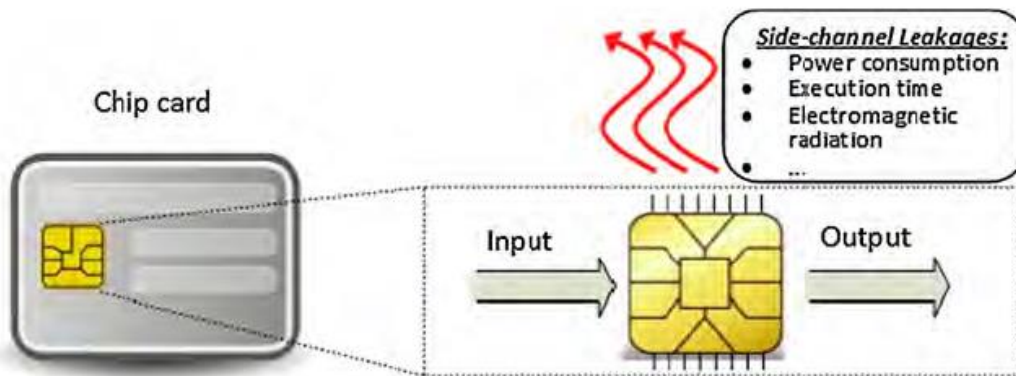
- The third class of design for trust is trustworthy computing on untrusted components. The difference between runtime monitoring and trustworthy computing is that trustworthy computing is tolerant to Trojan attacks by design.
- Some approaches employ a distributed software scheduling protocol to achieve a Trojan-activation-tolerant trustworthy computing system in a multicore processor. Concurrent Error Detection (CED) techniques can be adapted to detect malicious outputs generated by Trojans.

iv) Split-Manufacturing for Hardware Trust

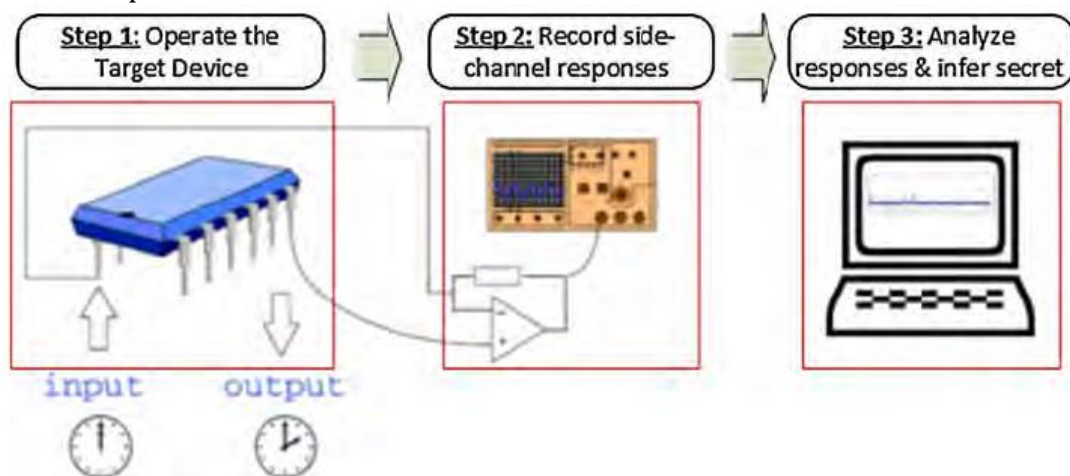
- Split-manufacturing has been proposed recently as an approach to enable use of state-of-the-art semiconductor foundries while minimizing the risks to an IC design. Split manufacturing divides a design into Front End of Line (FEOL) and Back End of Line (BEOL) portions for fabrication by different foundries.
- An untrusted foundry performs FEOL manufacturing (higher-cost), then ships wafers to a trusted foundry for BEOL fabrication (lower-cost).
- The untrusted foundry does not have access to the layers in BEOL and, thus, cannot identify the “safe” places within a circuit to insert Trojans.
- Existing split manufacturing processes rely on either 2D integration, 2.5D integration], or 3D integration.
- The 2.5D integration first splits a design into two chips fabricated by the untrusted foundry and then inserts a silicon interposer containing inter chip connections between the chip and package substrate. Therefore, a portion of interconnections could be hidden in the interposer that is fabricated in the trusted foundry. In essence, it is a variant of 2D integration for split manufacturing.
- During the 3D integration, a design is split into two tiers fabricated by different foundries. One tier is stacked on top the other, and the upper tiers are connected with vertical interconnects called TSVs. Given the manufacturing barriers to 3D in industry, 2D- and 2.5D-based split manufacturing techniques are more realistic today.

Side channel attacks

- Side-channel attacks (SCA) are a noninvasive attack that is based on targeting the implementation of a cryptographic algorithm rather than analyzing its statistical or mathematical weakness.
- These attacks exploit physical information leaking from various indirect sources or channels, such as, the target device's power consumption, electromagnetic (EM) radiation, or the time taken for a computation. These channels are referred to as "side channels".



- Figure illustrates how a device leaks side-channel information while operating. Common side channel attacks, such as power attacks, monitor the device's power consumption.
- While the device is in operation, the power consumption can be measured using an oscilloscope, and the relation between the power consumption and the secret key is analyzed in various ways.
- Simple power analysis (SPA) is a technique to directly interpret the collected traces of power consumption for a set of inputs. It requires relatively detailed knowledge about the implementation of a cryptographic algorithm and a skilled adversary to interpret secret key information by visually examining the power consumption.



- In contrast, differential power analysis (DPA) is a statistical analysis approach that does not require detailed knowledge of the target hardware implementation, which may be considered as a black box.

- DPA has been shown to be effective in finding a correlation between power consumption and processed data related to the secret key by statistical methods. In order to perform DPA successfully, however, often large number of power measurements are required.

TAXONOMY OF SIDE-CHANNEL ATTACKS

- Based on the level of control that an attacker may have on a device prior to performing SCAs, they can be classified into passive and active attacks.
- Passive attacks (such as power, timing, or EM SCAs) do not require an attacker to interfere with the functionality or the operation of the device under attack. The attack is usually launched in a manner that allows the system to behave normally as if the attack is not in effect.
- On the other hand, active attacks aim to interfere with the operation of the device under attack, where an attacker tends to influence how the device behaves, and what operation it performs.

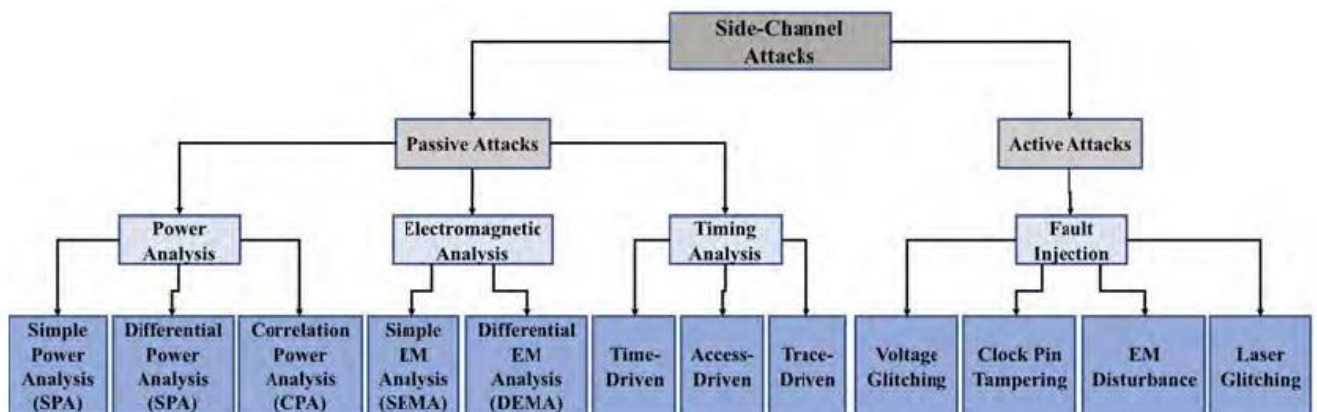


Figure shows the taxonomy of SCAs. Depending on the general source of side-channel information, there are several forms of SCA. They are: power SCA, EM SCA, fault injection attack, and timing SCA.

Each SCA can be classified according to specific attack method: applied analysis methods, such as simple observation and statistical methods; side-channel signal generation methods, such as voltages and clocks; or analysis granularity, such as microarchitecture and system level analysis.

UNCOMMON SIDE-CHANNEL ATTACKS

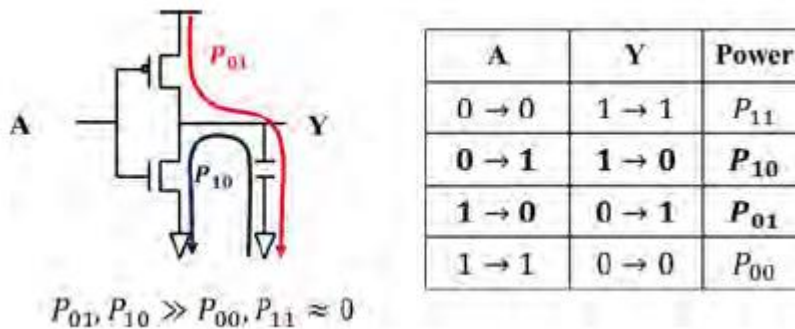
- Besides the common ones described earlier, there are several other side-channel signals that can leak information about stored secrets in hardware. These signals include emitted sound, temperature, and vibration.
- One example of these uncommon SCAs is acoustic side-channel analysis. The attack focuses on systems that produce sounds while being operated (such as, 3D printers), where program information can be extracted from the leaked acoustic signals.
- Other uncommon side-channels, such as, temperature and vibration, can also leak a significant amount of critical information about the device under attack.

POWER ANALYSIS ATTACKS

The basic idea of power analysis attacks is to reveal secret information from a device by analyzing its power consumption. Power analysis attacks are mainly used to extract the secret key of cryptographic systems, as it has been used to successfully break the Advanced Encryption Standard (AES) in a few minutes.

Dynamic power

- Two factors affect the power consumption of a device; the first factor is the dynamic power, which is caused by the switching activities of transistors within the device. The second factor is the leakage power, which is an unwanted behavior of a transistor, associated with the leakage current generated in its off state.
- An adversary is usually interested in capturing the dynamic power signals as they are directly related to the functional behavior of the device. For example, the dynamic power of an inverter is related to the switching activities of the input and the output, as shown in Fig.



- Let P_{ij} be the power consumption, where the output value of the inverter is changed from i to j , for $i, j \in \{0, 1\}$.
- P_{01} and P_{10} are much greater than P_{00} and P_{11} , since the capacitor connected to the output is charged or discharged when the output value is switched;
- P_{00} and P_{11} are almost zero since there is no charging or discharging activity.
- Based on this characteristic, an adversary can estimate the status of the output or input by measuring the power of an inverter.
- If the input of an inverter originates from a secret key, the power side-channel leakage gives an adversary a clue about the secret key.

ACQUISITION OF POWER SIGNALS

- The process of power acquisition requires basic knowledge about the functionality of a device, where an input pattern is applied, and the power traces are captured during the processing of those patterns.
- The power signals are captured by measuring the change in current levels in the voltage supply transmission lines. Usually, an oscilloscope measures the voltage drop across a precision sense resistor connected between the power rail and V_{dd} or Gnd pin of the target device.

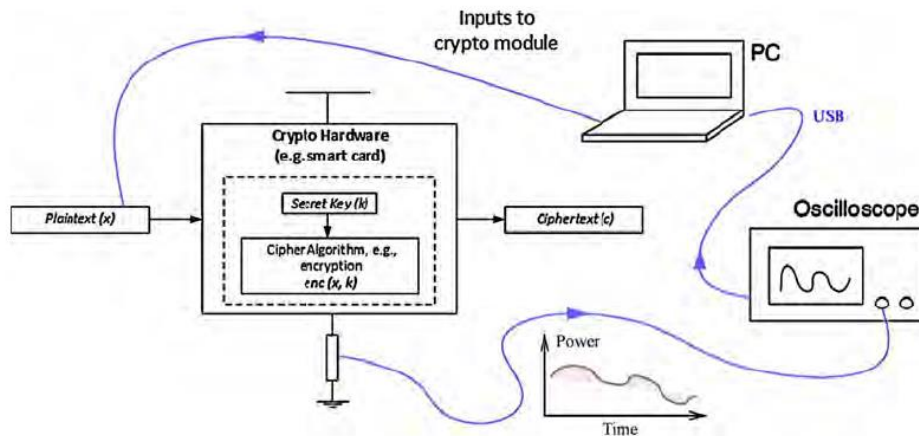


Figure presents an overview of a power analysis setup, where a cryptographic system is being controlled by a computer that applies input patterns, observes the outputs, measures the power consumption, and performs necessary analysis to extract the secret key.

TYPES OF POWER ANALYSIS

Three types of power side-channel analysis are simple power analysis (SPA), differential power analysis (DPA), and correlation power analysis (CPA).

Simple Power Analysis (SPA)

- SPA is a technique that aims to observe power measurements obtained while the device under attack is in operation mode.
- This type of analysis does not require any advanced or statistical processing stages.
- SPA attacks are usually applied to devices with limited accessibility, where one or few power traces are available.
- Visual inspection of power traces is considered the primary form of SPA attack, where a power trace shows a sequence of patterns that can lead to identifying key bits, instructions, or functions.
- Each instruction in a processor causes a specific pattern that can be visually identified in the power trace.
- Template attacks are a more advanced form of SPA, where known and recognized patterns in the power trace are characterized and stored as templates.
- Power traces collected from the target device are matched to the templates, and then corresponding operations are recognized.

Differential Power Analysis (DPA)

- DPA attacks are the most common type of side-channel attacks, due to the fact that attackers are not required to have prior knowledge about the hardware architecture of the device under attack to perform the analysis.
- Additionally, DPA has been proven very effective in obtaining high-quality signals in a noisy environment.
- Compared to SPA, DPA typically requires larger number of traces; more data collection makes DPA more powerful.

- Implementation of DPA attack requires two phases: data collection and data analysis. In the data collection phase, different input patterns are applied to the device while recording the power traces in a high sampling rate.
- Averaging the measured traces, and applying a band pass filter that is tuned to remove noise, can help improve the quality of traces.
- In the data analysis phase, statistical analysis, such as the difference of means is applied.
- High-order DPA is a technique that utilizes multiple points in the entire power trace, or high-order statistics of a point, such as the second, third, and higher moments (i.e., variance, skewness, and kurtosis).
- This technique can successfully exploit the device's vulnerabilities, and bypass the traditional power analysis countermeasures.
- The amount of information, the low cost, and the noninvasive nature of the attack make it one of the most powerful side-channel analysis attacks.

Correlation Power Analysis (CPA)

CPA is an advanced form of SCA that exploits the correlation between the power consumption, and the Hamming distance or Hamming weight of the target function, for example, the output of the SBOX operation.

The first step of the CPA attack is to determine the intermediate value of the cryptographic algorithm, that is the target function, which is denoted by $v_i = f(d_i, k^*)$, where d_i is the i^{th} plaintext or ciphertext, and k^* is the hypothesis of a component of the secret key.

The second step is to measure the power consumption of the cryptographic device while it encrypts or decrypts D , different data inputs, including the target function at the first step. We denote the power trace as $t_i = (t_{i,1}, t_{i,2}, \dots, t_{i,t^*}, \dots, t_{i,L})^T$, corresponding to input d_i , where L denotes the length of the trace, and t_{i,t^*} is the power consumption when the target function at the first step is performed.

The third step is to calculate a hypothetical intermediate value for all possible k : $v_{i,j} = f(d_i, k_j)$ for $i = 1, \dots, D$ and $j = 1, \dots, K$.

The fourth step is to map the hypothetical intermediate values to the hypothetical power consumption values: $h_{i,j} = g(v_{i,j}) = g(f(d_i, k_j))$ for $i = 1, \dots, D$ and $j = 1, \dots, K$.

The fifth step is to compare the hypothetical power consumption model with the measured power traces.

In order to measure the linear relationships between the two vectors h_i and t_j for $i = 1, \dots, K$ and $j = 1, \dots, T$, the correlation coefficient is calculated:

$$r_{i,j} = \frac{\sum_{d=1}^D (h_{d,i} - \bar{h}_i)(t_{d,j} - \bar{t}_j)}{\sqrt{\sum_{i=1}^D (h_{d,i} - \bar{h}_i)^2 \sum_{i=1}^D (t_{d,j} - \bar{t}_j)^2}}$$

where \bar{h}_i and \bar{t}_j denote the mean values of the vector \vec{h}_i and \vec{t}_j , respectively. If r_{k^*,t^*} of the correct key k^* and the specific time t^* has the distinct peak value, then the CPA attack is successful.

POWER SIDE-CHANNEL ATTACK COUNTERMEASURES

- In order to remove dependency between power consumption and intermediate values of the executed cryptographic algorithm, the cryptographic hardware can be implemented with secure primitive logic cells (such as sense-amplified-based

logic (SABL), wave dynamic differential logic (WDDL) , and t -private logic circuit) at the design stage.

- These secure logic styles use different methods to make the power consumption of the performed operation independent of the processed data values, thus preventing leakage of secret information (i.e., key) in power traces.
- SABL and WDDL consume equal amounts of power in each clock cycle, but t -private logic circuit randomizes amounts of power consumption in each clock cycle by masking each bit with t random bits.
- In other words, SABL and WDDL implement the hiding countermeasure, and t -private logic circuit implements the masking countermeasure.

	SABL	WDDL	t-private logic
SCA resistance	✓	✓	✓
Probing resistance	×	×	✓
Method	Hiding	Hiding	Random masking
Design	Full custom	Semicustom	Semicustom
Area	Medium	Low	High
Power	Medium	High	Low

Table shows the summary of these secure logic styles.

ELECTROMAGNETIC (EM) SIDE-CHANNEL ATTACKS

- EM SCA focuses on measuring electromagnetic waves that are emitted from ICs in operation.
- The EM waves are produced as current flows across a device, where transistor and interconnect switching activities occur with changing input patterns. This current flow results in the EM signals.
- An adversary usually aims to capture EM signals that are produced by current flows of data processing stages, where most waves occur, due to the switching activity of a device while performing a data processing operation. These waves are usually considered unintentional, and they allow critical information to be leaked naturally during operation.

EM EMANATIONS

EM emanation is defined as the process that causes the target device to generate EM signals. There are two types of EM emanations: intentional and unintentional emanations.

Intentional Emanations

- Intentional EM emanation results from current flows that are applied to cause the device to emit an electromagnetic response. These current flows are usually in the form of short bursts and sharp rising edges.
- The objective of an attacker in this type of emanation is to isolate the EM response of the targeted critical data path.

Unintentional Emanations

- When an adversary applies EM side-channel analysis, focusing on unintentional emanation can help identify critical paths and acquire their data values.
- The increased complexity and reduced size of modern ICs result in electric and electromagnetic coupling between components, which is an uncontrolled phenomenon that can generate a compromising signal.
- These components may act as modulators; they generate a carrier signal that can be intercepted and post-processed to acquire the carried data.

TYPES OF EM ANALYSIS

There are two main types of EM analysis: Simple and differential electromagnetic analysis, SEMA and DEMA, respectively.

Simple Electromagnetic Analysis (SEMA)

- In SEMA, an attacker obtains a single time domain trace to observe, and gain knowledge about the device directly. The attack is only valid when there is prior knowledge about the device's architecture, or the security policy, applied.
- The primary objective in SEMA is to obtain critical information via visual inspection of the EM signal trace, where a sequence of transitions at the startup of the system may include information about the secret key used to encrypt/decrypt data.
- The use of SEMA is usually the first step of EM SCA, where the necessary information can be observed to carry on a more detailed analysis using DEMA.

Differential Electromagnetic Analysis (DEMA)

- The attacker applies DEMA to the device to exploit information that cannot be visually observable. DEMA generally utilizes a self-referencing approach, which compares the analyzed signal with an equivalent one in a different area of the device (spatial referencing), or in a different time (temporal referencing).
- DEMA does not require much knowledge about the device under attack; most information can be exploited when obtaining different forms of EM signals in different places and times.
- The analysis in DEMA can help identify functional and structural details of the target device. It can also track a process flow and determine how a signal propagates inside the device.
- These details obtained by DEMA can help reverse engineer the device, or give the attacker the ability to disable the security policy of the system physically.

EM SCA COUNTERMEASURES

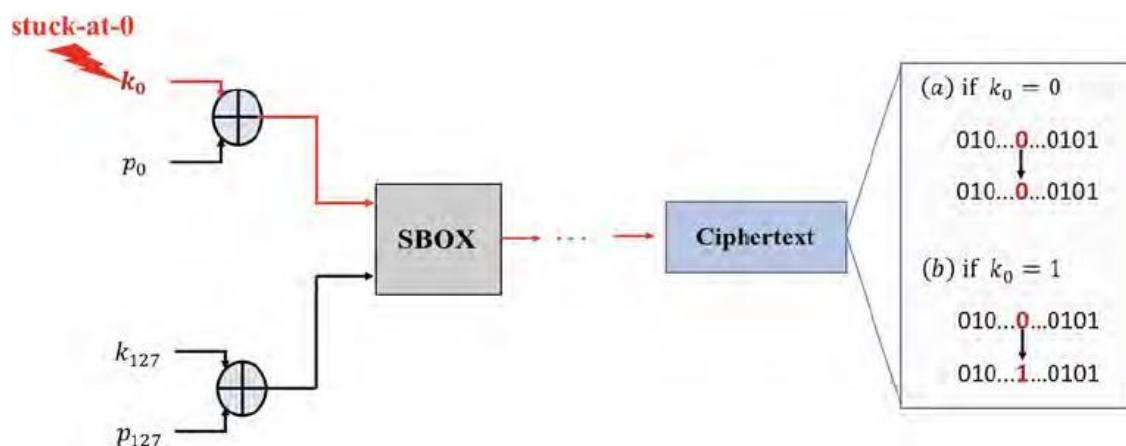
- To protect against EM SCAs, many countermeasures can help add a layer of protection, while maintaining the device's performance and quality of service.
- Redesigning the circuit to reduce the coupling issue is one of the primary countermeasures.
- Additionally, adding a layer of shielding to the device to prevent EM signals from propagating is another significant measure.
- Introducing nonfunctional modules that produce EM noise can also prevent critical information from being easily intercepted due to the high amount of noise being applied in the same frequency band.

- Other functional countermeasures may hide critical processes from being detected, such as introducing a crucial nonlinear processing sequence when using a cryptographic system. By injecting dummy instructions or operations between the stages of a cryptographic process, the adversary can be prevented from differentiating between key-bits and dummy-bits, even when performing successful EM side-channel attacks.

FAULT INJECTION ATTACKS

- Fault injection attacks are active attacks, where a crypto-device is intentionally injected with a fault that leads to a leakage of the secret key.
- A typical attack starts with injecting a fault to an operational device, which can either be a voltage or a clock source glitch. Other techniques, such as electromagnetic radiation, or physical probing, or passing a laser beam through the device can also be used for the fault injection.
- The device is then analyzed by observing the faulty outputs; these outputs can potentially help extract the secret key.
- The attack is considered semi-invasive, as the physical modification is sometimes needed for the fault to be properly injected.

An example of a simple fault injection attack on AES is shown in Fig.



- Let k_0 represent the bit 0 of the AES key. Now, consider that a stuck-at-zero fault is injected in k_0 during the initial Key Addition operation, which forces the value of k_0 to be zero.
- There are two possible outcomes. First, if $k_0 = 0$ prior to fault injection, then the fault induced has no effect on the output. On the other hand, if $k_0 = 1$, then the fault toggles the value of $p_0 \oplus k_0$.
- This is a disturbance, which then gets propagated as execution progresses, resulting in a faulty ciphertext. In this case, the faulty ciphertext will be different from that of the fault-free ciphertext, generated from the same plaintext.
- The attacker, thus, identifies the value of k_0 .
- In a similar way, by independently injecting faults in key bits k_1 to k_{127} , the attacker can retrieve the entire AES key.

FAULT INJECTION TECHNIQUES

Fault injection techniques can vary based on the type of device, and the amount of information available to the attacker.

1. Voltage Glitching

- Voltage glitching is considered the basic fault injection technique, where a device is supplied with a lower-than-normal voltage level.
- By running the device in this state, faults would start to appear at the output of the device.
- The precision and type of fault are controlled by the level of the supplied voltage; a more faulty behavior is obtained as the voltage supply is lowered further.
- This technique is not invasive and it does not require specific timing patterns. When applying this attack, faults will propagate uniformly across the device, which can give an advantage to the attacker in terms of accessibility to rarely-activated nodes and registers in the device.

2. Tampering With Clock Pin

- Another basic fault injection technique involves an attacker subjecting the device with a faulty clock signal. The fault can either be a glitch in the clock, or the voltage level of the signal.
- This noninvasive attack forces the device to generate faulty outputs across all clocked operations.
- The attacker needs to have access to the clock signal of the device in order to precisely control the fault, which is a relatively easy task.

3. EM Disturbances

- EM Disturbances can be applied to a device to inject faults. Generating EM signals and directing them to a device can cause the operation of the system to be compromised.
- By controlling the EM signal, different types of behavioral changes in functionality can be observed, and with the right input patterns and enough iteration, the device can leak the secret key of the cryptographic module.
- Since EM signals are applied to the entire device, the attack affects the system uniformly, as the faults can be injected at any location in the device.

4. Laser Glitching

- Applying a laser beam to a specific area of a device can cause a fault to be injected. Data in registers and states can be modified when intentionally applying a strong laser beam.
- Beams can be controlled in terms of strength and polarization, which allows the attacker to either inject faults to a specific area, or to the entire device.
- Injected errors can propagate to the output of the design, and successfully leak the secret key of the cryptographic module.

FAULT INJECTION COUNTERMEASURES

- One of the most common fault injection attack countermeasures is based on replication of critical operations, which is a popular solution for fault-tolerant computing.
- Here, the crypto operations are repeated, and the two outputs are compared. If found different, the system assumes that a fault has been injected, and appropriate actions are taken.
- The replication can be done spatially or temporally. Spatial replication, especially applicable in hardware crypto accelerators, has redundant circuit blocks to re-compute specific crypto operations.
- Temporal replication reuses the same circuit blocks to perform the re-computation at different time.
- Whereas the spatial countermeasure does not affect the execution time for crypto operation (but adds area overhead), the temporal countermeasure does not affect the area requirements (but adds delay overhead).
- The former is, therefore, suited for high-speed applications, whereas the latter is suited for small devices.
- An alternate protection approach is based on error-detection schemes, such as parity checks. The scheme adds a detection mechanism that disables the critical functionality of the device when it is operating in a faulty environment.
- Compared to replication, they have typically less overheads. However, they are not very efficient in detecting multiple fault injections. Overhead goes up significantly with the need to detect and/or correct multiple faults.
- Anti-tamper protection modules are also an option. Anti-tamper protection modules can act as scanning tools that look for and report any physical modification attempts. These modules are limited to physical and semi-invasive attacks.

TIMING ATTACKS

Timing analysis is an SCA that is used to extract critical information about the device under attack by analyzing the execution time of each operation under different setups and input patterns.

TIMING ATTACKS ON CRYPTOGRAPHIC HARDWARE

- An adversary often applies timing analysis on cryptographic systems to extract the secret key, where timing analysis can help the attacker determine which subsets of the key are correct, and which subsets are not.
- The way an adversary measures the delay of a signal is by applying a change in the input, and recording the delay that occurs before the output is updated.
- Other techniques include focusing on power or EM signals to analyze the delay; this is mainly used when the device under attack has a sequential circuit, or uses pipelines.
- Environmental conditions may help an adversary perform effective timing analysis, where different operating temperatures may affect the speed of data flow. For example, higher temperature typically causes a slower data flow, which can help differentiate between parallel or high-speed operations.

- Timing attacks are usually applied along with other side-channel attacks, since more information can be extracted when different analysis methods are employed.
- Power analysis is one example that works well with timing attacks; the power trace does not only show the pattern in which the operation performed is correlated to, but also how long it took before the operation is completed.
- The order of operation is also revealed when applying timing analysis to power signals; this order can help identify the type of process the device is running, and may even allow the adversary to reverse engineer the device.

CACHE-TIMING ATTACK IN PROCESSOR

- Another powerful timing analysis attack, called cache-timing attack, is applied to the cache memory of a processor.
- The main objective of the cache-timing attack is to measure the time for cache access and then to relate the timing values to the information being processed.
- The cache access time is different for the following two cases: (1) when the requested data by the processor is available in the cache (i.e., cache hit), and (2) when the cache does not have the data available and requests it from the main memory (i.e., cache miss).
- Retrieving data from the main memory, or from cache levels closer to the main memory in the memory hierarchy of a processor, takes longer time than that from cache levels closer to the processor core. These timing differences have been exploited in SCAs.
- To measure the time, an attacker can flush the monitored memory line from the cache hierarchy (FLUSH phase), and then wait to allow the victim program to access the memory line (WAIT phase).
- An attacker then reloads the memory line, measuring the time to load it (RELOAD phase). If the victim accesses the memory line during the wait phase, the reload operation takes a shorter time. Otherwise, the requested line needs to be brought from the memory, and the reload takes significantly longer time.
- Figure shows the timing of the attack phases with and without the victim access. This attack is called Flush+Reload attack.

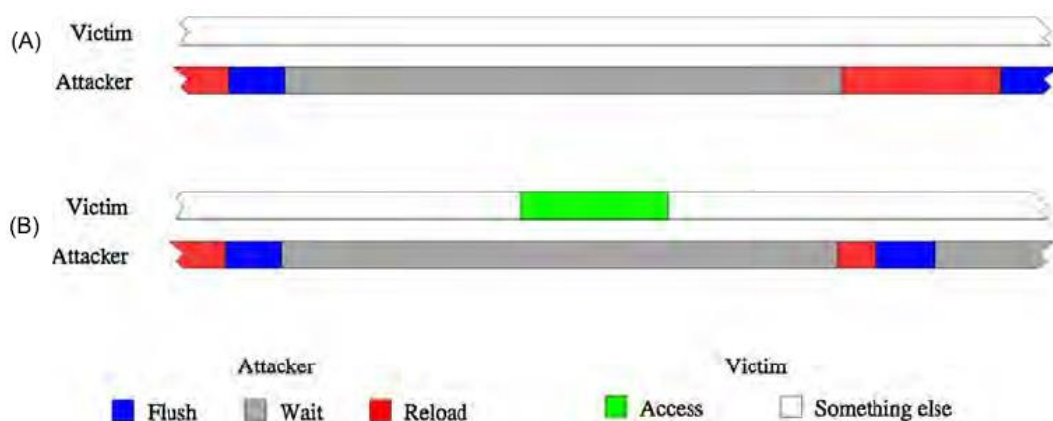


FIGURE 8.13

Timing of Flush+Reload attack: (A) without victim access; (B) with victim access [23].

TIMING ATTACKS COUNTERMEASURES

- To protect devices against timing attacks, designers can do the following:
 - Randomize the delay of different operations, or
 - Make all operations take the same time, thus preventing information leakage through timing channel.
- While constant-time implementations can guarantee security against timing attacks, they are not easily achievable in practice.
- Randomization on the other hand, for instance, by adding random delays to the execution of a task, is easier to accomplish. While it makes the attack more difficult, it cannot, however, guarantee the security of an implementation against timing attacks.
- Randomization is done by creating various execution paths and adding different delays to different paths.
- One way to apply delay to a path is to place a series of buffers in the path during circuit design, where the number of buffers can be controlled by the designer to maintain the desired delay.

Table 8.2 Summary of side-channel attacks

Side-Channel Attack	Measured Parameters	Analysis Methods	Countermeasures
Power Analysis	Current signature and power consumption patterns	Simple power analysis (SPA) Differential power analysis (DPA) Correlation power analysis (CPA)	Power consumption masking Power consumption hiding
EM Analysis	Intentional and non-Intentional electromagnetic emission	Simple EM analysis (SEMA) Differential EM analysis (DEMA)	EM emission shielding EM noise generation modules
Fault Analysis	Invalid outputs, underpowered behavior, and Laser/UV Glitching Responses	Comparative approach to analyze responses before and after fault insertion	Error detection schemes Anti-tamper protection modules
Timing Analysis	Operation delays, time elapsed when different input patterns are applied	Analysis to relate operation delay to the nature of the function	Randomized operational delay Fixed operational delay

Physical attacks and countermeasures

- Physical attacks are divided into three categories: noninvasive, semi-invasive, and invasive attacks.
- A noninvasive attack does not require any initial preparations of the device under test, and will not physically harm the device during the attack. The attacker can either tap the wires to the device, or plug it into a test circuit for the analysis.

- Invasive attacks require direct access to the internal components of the device, which normally requires a well-equipped and knowledgeable attacker to succeed.
- Meanwhile, invasive attacks are becoming constantly more demanding and expensive, as feature sizes shrink, and device complexity increases.
- Many attacks fall into this gap, called semi-invasive attacks. They are not very expensive as classical penetrative invasive attacks, but are as easily repeatable as noninvasive attacks.
- Like invasive attacks, they require de-packaging the chip in order to get access to its surface.
- Some physical invasive attacks are Reverse engineering micro probing attack, and invasive fault injection attack.

1. Reverse Engineering

- Reverse engineering (RE) is the process involving the thorough examination of an object to achieve a full understanding of its construction and/or functionality; a method used by attackers as part of mounting their attack.
- RE is now widely used to clone, duplicate, or reproduce systems and devices in various security-critical applications, such as smartcards, smartphone, military, financial, and medical systems.
- Anti-RE techniques should have the ability to monitor, detect, resist, and react to invasive and noninvasive attacks.

Following is a presentation on the RE of electronic devices from chip to system levels:

A) CHIP-LEVEL RE

- An IC typically consists of a die, a lead frame, wire bonding, and molding encapsulant. The package of a chip can be classified in different ways. The materials that are used can be ceramic or plastic. Packaging can also be wire bond or flip chip.
- At the chip level, the goal of the RE process is to find package materials, wire bonding, different metal layers, contacts, vias and active layers, and interconnections between metal layers.

The RE process has several different steps:

Decapsulation: Decapsulation exposes the internal components of the chip, which allows for the inspection of the die, interconnections, and other features.

Delayering: The die is analyzed layer by layer, destructively, to see each metal, passivation, poly, and active layer.

Imaging: An image is taken of each layer in the delayering process by using scanning electron microscope (SEM), Transmission electron microscope (TEM), or Scanning capacitance microscopy (SCM).

Post-processing: In this process, the images from the previous step are analyzed, schematic and high-level netlists are created for functional analyses, and the chip is identified.

Decapsulation

- First, reverse engineers identify the package materials and remove the chip's packaging. Depot is the traditional method by which an acid solution is used for removing the package. These acid solutions are used to etch off the packaging material without damaging the die and interconnections.

- To remove the die package, one can use selective or nonselective methods. Wet chemical etching and plasma etching can be used as selective techniques, whereas nonselective techniques would be thermal shock, grinding, cutting, and laser ablation.
- After decapsulation, the die needs to be cleaned before delayering and/or imaging can be performed, because dust may be present, resulting in artifact. Different methods for cleaning the dust are Spray cleaning, Acid cleaning, Ultrasonic cleaning, Mechanical swabbing.

Delayering

- Modern chips are made up of several metal layers, passivation layers, vias, contact, poly, and active layers. Reverse engineers must perform cross-section imaging of a chip, using SEM or TEM to identify the number of layers, metal material, layer thickness, vias, and contacts.
- Several methods can be used simultaneously when a chip is delayered, such as wet/plasma etching, grinding, and polishing. A reverse engineer should determine the etchants needed, and the time required to remove each layer.
- Once the etchants are determined for delayering a specific layer and metal, a reverse engineer will begin by etching the passivation layer; then, the reverse engineer will take an image of the highest metal layer; after that, the reverse engineer will etch the metal layer. This same process is repeated for each layer, including the poly and active layers.

Imaging

During the delayering process, thousands of high-resolution images are taken to capture all of the information contained in each layer. Later, these images can be stitched together, and then studied to recreate the chip. For the purposes of imaging, many high-resolution microscopes and x-ray machines could be used.

Post-Processing

The post-processing or circuit extraction after delayering consists of the following steps: (1) image processing, (2) annotation, (3) gate-level schematic extraction, (4) schematic analysis and organization, and (5) high-level netlist extraction from the gate-level schematic.

1) Image Processing.

Taking images manually is becoming increasingly difficult, because the size of the ICs is shrinking, along with many of their features. Advanced electrical labs now use automated instruments (x-rays, SEMs, digital microscopes) that are equipped to take images of entire layers of ICs and PCBs. Then, the automated software can be used to stitch the images together with minimal error, and synchronize the multiple layers without misalignment.

2) Annotation.

After the completion of the aligned layers and stitched images, the extraction of the circuit starts. This stage in the process includes making note of transistors, inductors, capacitors, resistors, diodes, other components, the interconnection of the layers, vias, and contacts. The circuit extraction could be an automated or a manual process.

3) Gate-level schematic extraction.

Sometimes the images are imperfect, as the images may be taken manually. Additionally, the annotation process and image recognition for digital cells could be

erroneous. Therefore, verification is needed before the creation of a schematic. Design rule checks could be used to detect any issues related to minimum-sized features or spaces, wire bonding, vias, and connections.

4) **Schematic analysis and organization.**

The schematic analysis should be done thoughtfully and carefully with proper hierarchy and design coherence. For the analysis and organization of a schematic, the reverse engineer could use public information on the device, such as its datasheet, technical report, marketing information, and patents. This could help to facilitate an analysis of the architecture and circuit design.

5) **High-level netlist extraction from gate-level schematic.**

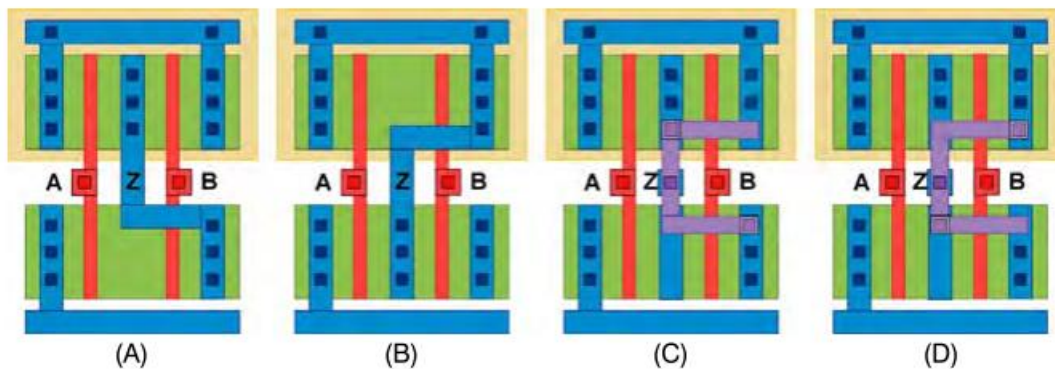
After circuit extraction is performed on the stripped IC (derivation of circuit schematic diagram), several techniques could be applied to get the high-level description for analysis and validation of the functionality of the chip, using simulation.

CHIP-LEVEL ANTI-RE

There are several approaches for the anti-RE of ICs, which include camouflage, obfuscation.

Camouflage

In the camouflage technique, the layout of standard cells with different functionalities is made to appear identical. One can introduce camouflage to a standard gate by using real and dummy contacts, which can enable different functionalities, as shown in Fig. In Figs. A and B, the layouts of two-input, NAND and NOR gates, are shown. These gates functionalities can be easily identified by their layouts. In contrast, Figs C and D show camouflaged two-input NAND and NOR gates with layouts that appear identical. Camouflage could be used to hinder adversaries who want to perform RE on a chip.



Obfuscation

Obfuscation techniques entail making a design or system more complicated to prevent RE, while also allowing the design or system to have the same functionality as the original. There are several different obfuscation approaches. The hardware protection through obfuscation of netlist could be used against piracy and tampering, and the technique could provide protection at every level of the hardware design and manufacturing process.

B) BOARD-LEVEL RE

The goal of board-level RE is to identify all components on the board and the connections between them.

IC identification via chip and die markings.

Some electronic components mounted on the PCB can be identified easily through the use of IC markings, but fully custom and semicustom ICs are difficult to identify.

IC markings can be divided into the following four parts

- The first is the prefix, which is the code that is used to identify the manufacturer.
- The second part is the device code, which is used to identify a specific IC type.
- The next part is the suffix, which is used to identify the package type and temperature range. Manufacturers modify their suffixes frequently.
- A four-digit code is used for the date, where the first two digits identify the year and the last two identify the number of the week. In addition, manufacturers could cipher the date into a form only known by them.

After identifying the manufacturer and IC markings, the reverse engineer could find the detailed functionality of the chip from the datasheets, which are available on the Internet. If the IC marking is not readable, the reverse engineer could strip off the package, and read the die markings to identify the manufacturer and the chip's functionality.

Destructive analysis of PCBs.

- Before PCB delayering, images of the placement and orientation of all outer layers' components are captured. Then the components could be removed, drilled hole positions could be observed, and it could be determined whether there are any buried or blind vias.
- After the PCB is delayered, images of each layer can be taken. Then the composition and the thickness of the layers should be noted. It is important to track the impedance control of high-speed signals and the characteristics of the PCB. The dielectric constant, prepreg weave thickness, and resin-type should also be determined.

Nondestructive 3D imaging of PCBs using x-ray tomography.

- X-ray tomography is a noninvasive imaging technique that makes it possible to visualize the internal structure of an object without the interference of over-and under layer structures.
- The principle of this method is to acquire a stack of 2D images, and then use mathematical algorithms such as the direct Fourier transform and center slice theory to reconstruct the 3D image.
- These 2D projections are collected from many different angles, depending on the quality needed for the final image.
- The object properties, such as dimension and material density, source/detector distance to object, source power, detector objective, filter, exposure time, number of projections, center shift, and beam hardening are important to consider in the selection of the tomography process parameters.

Netlist extraction after imaging.

After capturing images of the PCB via delayering or x-ray tomography, connections between all of the components could be discovered, which would yield a PCB layout netlist.

To create the netlist from the collected images, one should verify the following:

- Connection between the components of original board (a datasheet could be helpful to find the connection for original functionality),
- Unexpected shorts and hanging Vdd,
- Pin connections between components.

BOARD-LEVEL ANTI-RE

Ensuring complete protection from PCB-level RE is a difficult task, and thus the goal of anti-RE methods is to simply make RE prohibitively expensive and time consuming. A summary of PCB-level anti-RE techniques are:

1. Tamper-proof fittings (such as t or x), custom screw shapes, adhesively bonded enclosures, and fully potting the space around a PCB could be used for protection against physical attacks.
2. Custom silicon, unmarked ICs, missing silkscreens with minimum passive components, and a lack of information from the Internet could complicate RE. Additionally, the elimination of JTAG and debug ports from silicon can make the RE process harder.
3. Ball grid array (BGA) devices are better, because such devices do not have exposed pins. Back-to back BGA placement in a PCB board could be most secure. For back-to-back BGA placement, the PCB needs to be multilayered, which will increase the RE cost for layer-by-layer analysis. The problem is that back-to-back BGA packaging is complex and expensive.
4. If the devices are operating in an unusual fashion (for example, if there are jumbled addresses and data buses), then, it would be hard to find the functionality of the device. Obfuscation (that is, wiring connections between used pins to unused pins, having spare inputs and outputs from processors to route signals, dynamically jumbling buses, and jumbling the PCB silkscreen annotations) could complicate the RE process. However, such techniques also require the use of more complex chips, and complicated design methods.

C) SYSTEM-LEVEL RE

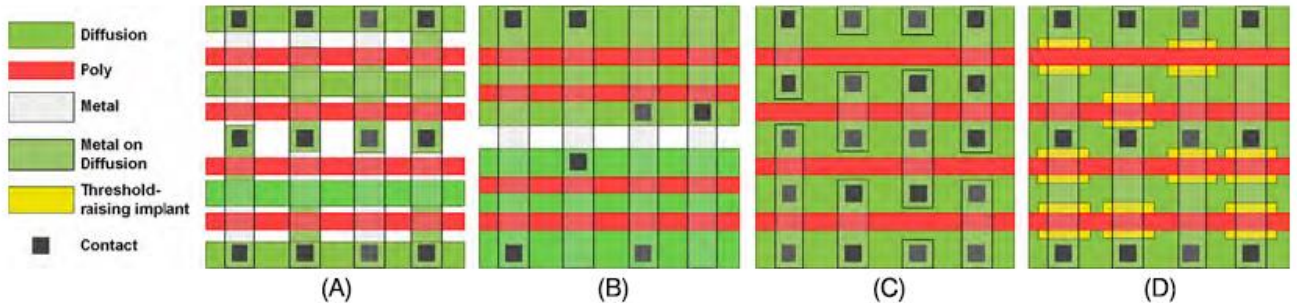
- With chip- and PCB-level RE processes, the purpose is to obtain the netlist of the chip and board in the embedded system, which represents the function and interconnections of the design. To make the design fully functional, the system operation codes and control instructions, which are defined by firmware, should be retrieved as well. This is referred as system-level RE.
- Firmware and netlist information can be stored via read-only memory (ROM), electrically erasable programmable ROM (EEPROM), or Flash memory.

ROM

ROM is a type of memory, whose binary bits are programmed during the manufacturing process. ROM devices can be typically classified into four types:

- ❖ Active-layer programming ROM: The logic state is represented by the presence or absence of a transistor. Fig A
- ❖ Contact-layer programming ROM: A bit is encoded by the presence or absence of a via, which connects the vertical metal bit line. Fig B
- ❖ Metal-layer programming ROM: The binary information is encoded by short circuiting the transistor, or not. Fig C

- ❖ Implant programming ROM: The different logic state is achieved by different doping levels in the diffusion area. Generally, higher doping levels will raise the on/off voltage threshold, which will disable the transistor. Fig D



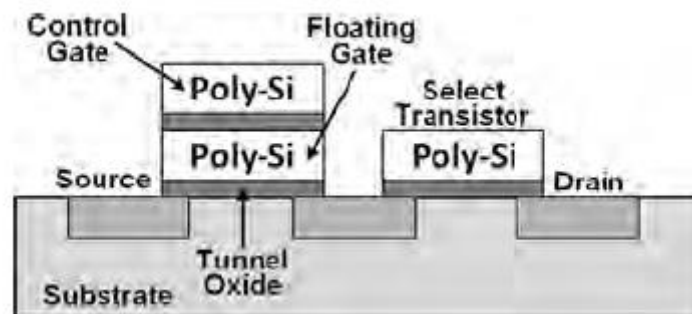
ROM RE

To reverse engineer the ROM content, one can take advantage of modern optical and electron microscopy to observe the binary states of each cell, as indicated below:

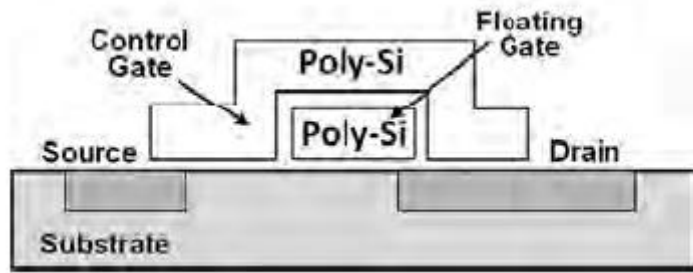
- ❖ Active-layer programming ROM: The metal layer and poly layer need to be removed using the delayering.
- ❖ Contact-layer programming ROM: It is much easier to reverse engineer this kind of ROM, as there is often no need to delayer the metal layer and the poly layer.
- ❖ Metal-layer programming ROM: This type of ROM can be directly observed under a microscope without having to perform any delayering process.
- ❖ Implant programming ROM: This type of ROM is inherently resistant to optical microscopy, as different logic states appear identical. To observe the impact of different doping levels, additional dopant-selective crystallographic etch techniques should be utilized to separate the two logic states.
- Generally, ROM only provides limited protection against RE. Among all types of ROM, the metal-layer programming ROM offers the worst security, because the metal layer is easy to obtain with little effort, whereas the implant programming ROM provides the highest level of protection available.

EEPROM/Flash Memory

One bit cell of EEPROM is composed of two transistors: floating gate transistor (FGT) and select transistor (ST). The FGT is feathered with two stacked gates: a control gate (CG) and a floating gate (FG). The logic state of the bit cell is encoded in the FGT by the presence or absence of electrons stored in the FG.



Flash memory has almost the same structure as EEPROM, except for the absence of ST.



EEPROM/Flash RE

Since EEPROM and Flash memory have similar structures and the same logic storage mechanism (as discussed earlier), they often can be reverse engineered by the same procedures. For quite long time, the EEPROM/Flash technology has been regarded as the most robust memory defense against RE. Recently, several methods, although very expensive and requiring specialized equipment, were proposed to extract the contents in EEPROM/Flash correctly.

Scanning Kelvin probe microscopy procedure

The first step is to remove the silicon from the back side of the memory, and leave the tunnel oxide layer undamaged to avoid charging/discharging of the FG. Then, the bit value can be read under the SKPM scan by applying a DC voltage to the probe tip which measures potential difference between the tip and the memory cell.

Scanning capacitance microscopy (SCM) procedure.

The SCM procedure measures the capacitance variations between the tip and the high-sensitivity capacitance SCM's sensor. The SCM sensor will detect the logic states via probing the carrier (hole) concentration. Thus, the back-side delayering should keep a silicon thickness of 50 to 300 nm to leave the transistor channel undamaged.

Property	SKPM Procedure	SCM Procedure
Delayering position	Back side	Back side
Delayering depth	Entire silicon	50–300 nm thickness
Sensitivity	Low	High
Measured carriers	Electrons	Holes
Measured parameter	Potential	Capacitance
Operation mode	Noncontact	Contact
Application	All EEPROM and some Flash	All EEPROM and Flash

FPGA

- An FPGA bitstream is essentially a vector of bits encoding the netlist information in FPGA, which defines hardware resources usage, interconnection, and initial states at the lowest level of abstraction.
- The logic blocks are configured to represent the basic digital circuit primitives, such as combinational logic gates and registers.
- The connection blocks and switch blocks are configured to be the interconnections between different logic blocks.

- Other hardware resources, such as I/O buffers, embedded RAM, and multipliers, can be programmed according to different requirements.
- Therefore, all information about the netlist can be obtained from the bitstream file.

RE of FPGAs

- FPGA RE involves analyzing the configuration bitstream file and transforming the bitstream file into the hardware netlist, which consists of all components and interconnections at the RTL.
- To fulfill this goal, hackers need to go through the following steps: get access to the bitstream file from the Flash memory, decrypt the bitstream (if encrypted), and finally build the mapping relationship between the bitstream file and the netlist.

Bitstream Access.

SRAM-based FPGA stores the logic cells states in the SRAM, which cannot retain the data after power loss. Therefore, an external NVM device (typically Flash) is adopted to hold the configuration bitstream file and transfer the bitstream file at system boot-up to initiate the SRAM in FPGA. The separation between the bitstream file and FPGA makes it easy to dump the contents of the bitstream file. By using a logic analyzer, one can easily wiretap the JTAG data and command lines to capture the communication between the FPGA and Flash memory during startup.

Bitstream Decryption.

To increase the security level of FPGA, most FPGA manufacturers encrypts the bitstream file before storing it in the Flash memory with the encryption standards, such as triple data encryption standard (DES) and advanced encryption standard (AES). Now the wiretapped encrypted bitstreams will not yield any information for RE, as long as the cryptographic key remains hidden inside the FPGA.

The bitstream decryption process in FPGA RE depends entirely on the attacker's ability to discover the key. Recently, the bitstream encryption of several mainstream FPGA series is vulnerable to the side channel attacks (SCAs). The leaked timing and power consumption information is collected when the encrypted bitstream is decrypted by the dedicated hardware engine within the FPGA. By analyzing the collected power consumption and timing behavior, the hypothetical structure of the internal triple DES module can be verified. Finally, the divide-and-conquer approach is applied to guess and verify a small portion of the key, which reduces the computation's complexity. This process is repeated until the entire key is obtained.

Bitstream Reversal.

A bitstream file consists of four parts: command header, configuration payload, command footer, and startup sequence. In the case of the Xilinx FPGA, the configuration payload determines the configuration points (such as LUT, memory, register, and multiplexer) and the programmable interconnection points (switch box). The goal of the bitstream reversal is to find out the mapping relationship between the configuration payload with the configuration points, and the programmable interconnection points.

Partial bitstream reversal.

This kind of bitstream reversal only focuses on extracting some specific configurable blocks in FPGA, such as the LUT, configurable logic block, and multiplier from the bitstream file.

Full bitstream reversal.

It makes the first public attempt to convert the bitstream file into the netlist. The set-theoretic algorithm and cross-correlation algorithm were used to build a database linking the bitstream bits to the associated resources (configuration points and programmable interconnect points) in the FPGA. Then, the database is utilized to produce the desired netlist based on any given bitstream file in Xilinx Virtex-II, Virtex-4 LXT, and Virtex-5 LXT FPGAs.

SYSTEM-LEVEL ANTI-RE

Anti-RE for ROMs

The most effective solution for increasing the complexity and difficulty of RE against ROM is to use the camouflage method.

Camouflage Contacts.

The camouflage contacts act as false connections between the metal layer and active layer to make the true contacts and the false contacts indistinguishable under optical microscopy. To decode the contents, careful chemical etching has to be applied to find the real contacts, and this is time consuming.

Camouflage Transistors.

To improve the security of active-layer programming ROM, false transistors are made to confuse the RE attempts, instead of using the absence of transistors. The false transistors, essentially with no electrical functions, have the same top-down view as the true transistors under an optical microscope. To crack the information, the attackers have to use more advanced electrical microscopes to analyze the top view and even the cross-sectional view of the ROM, which is usually economically prohibitive.

Camouflage Nanowires.

Through the use of nano material, ROM cells are fabricated within the vertical connections between the bit lines and the word lines of a ROM array. The real connections between bit lines and word lines act as transistors, whereas the nonelectrical dummy connections only play the role of design camouflage. Due to the small dimensions of the nanowires, the tiny differences between the dummy connections and real connections are indiscernible, even under advanced electrical microscopy. The major challenge with camouflage nanowires, however, is to manufacture the ROM at high enough volume and yield.

Anti-RE for EEPROMs/Flash

The most effective countermeasure would be to prevent back-side attacks. Some back-side attack detection methods are

Circuit parameter sensing.

Performing the delayering process from the back side will thin the bulk silicon. By burying two parallel plates in the bulk silicon to form a capacitor, the capacitance sensing can detect the capacitance reduction when the attacker polishes from the back side. When the capacitance reaches below a certain threshold, it will trigger the EEPROM/Flash memory to activate an erase operation. The capacitor, perpendicular to the bulk silicon, was previously a challenge to achieve. Fortunately, the emergence of the through-silicon via technique makes it much easier to fabricate. Similarly, other parameters, such as resistance, can be measured and compared to the predefined reference resistance threshold.

Light Sensing.

By optically monitoring the back side of the chip, the light-sensing method will equip at least one pair of light-emitting and light-sensing devices in the front side of chip, and light reflection module at the bottom of the silicon bulk. The light-emitting device is configured to emit light, which can penetrate the bulk, be reflected by the light reflection module, and then be collected by the light-sensing device. Once the delayering is applied, the changes in light distribution at the light sensing device can trigger the self-destruction of the data contained in the memory. This method can certainly make the RE process more time consuming. However, the costs associated with manufacturing and the power consumption from continuous light emitting, and sensing, make it less attractive in practice.

Ferroelectric RAM memory.

Ferroelectric RAM (FeRAM) has been shown to be a promising candidate for replacing EEPROM/Flash memory. FeRAM stores data by the polarization states of molecules. Due to the special state representations, the difference between two states under optical and electrical inspection is invisible.

Anti-RE for FPGAs

FPGA anti-RE techniques are categorized into three groups according to the FPGA RE procedure: bitstream hiding, side-channel resistance, and bitstream anti-reversal.

Bitstream hiding.

By integrating the bitstream storage memory with FPGA, the Flash FPGA and anti-fuse FPGA do not require external configuration memory, leaving the direct wiretapping useless.

Side-channel resistance.

The recent success of SCAs on the FPGA proves that the leakage of information poses a large threat to FPGA security. Thus, it is necessary to develop the side-channel resistance designs to protect the cryptographic keys. Intuitively, the most effective side channel resistance design is to remove the dependency between deciphering operations and power consumption. Another group of side-channel resistance designs can be found in the noise addition group. By introducing random power noise to make the power consumption of decryption nondeterministic, it is quite difficult for the attacker to determine which part of the power consumption is from the decryption.

Bitstream anti-reversal.

Until now, full bitstream reversal has only been theoretically possible. As one can imagine, the invasive attacks in the future may successfully find out the entire mapping between the encoding bits from the bitstream file and the hardware resources in the FPGA. FPGA vendors should study potential countermeasures to impede bitstream reversal under noninvasive attacks.

Another consideration is partial configuration. The critical configuration bits in the bitstream file (such as the IP core) are stored in the Flash memory within the FPGA, whereas other noncritical parts are still loaded from the external memory. This partial configuration only leaves the wiretapper partial information about the whole FPGA mapping information, thereby fundamentally eliminating the potential of bitstream reversal.

2. PROBING ATTACK

Physical attacks are capable of bypassing the confidentiality and integrity provided by modern cryptography through observation of a chip's silicon implementation. Probing

directly accesses the internal wires of a security-critical module and extracts sensitive information in electronic format. Probing poses a serious threat to mission-critical applications, and thus demands development of effective countermeasures from the research community.

Probing Attack Targets

It is essential for both attackers and countermeasure designers to determine which signals are more likely to be targeted in a probing attack. Such signals are termed as assets. Here a few examples that are the most likely targets for probing attacks are enumerated.

Keys: Keys of an encryption module (for example, private key of a public key algorithm) are archetypal assets. They are usually stored in nonvolatile memory on the chip. If the key is leaked, the root of trust it provides will become compromised, and could serve as a gateway to more serious attacks.

Firmware and configuration bitstream: Electronic intellectual properties (IPs), such as low-level program instruction sets, manufacturer firmware, and FPGA configuration bitstreams are often sensitive, mission critical, and/or contain trade secrets of the IP owner. Once compromised, counterfeiting, cloning, or exploits of system vulnerabilities could be facilitated.

On-device protected data: Sensitive data, such as health and personal identifiable information, should be kept private. Leakage of such information could result in fraud, embarrassment, or property/brand damage for the data owner.

Device configuration: Device configuration data control the access permissions to the device. They specify which services or resources can be accessed by each individual user. If the configurations are tampered with, an attacker could illegally gain access to resources to which, otherwise, he/she had no access.

Cryptographic random number: Hardware generated random numbers, such as keys,, onetime pads, and initialization vectors for cryptographic primitives also require protection. Compromising this type of asset will weaken the cryptographic strength of the digital services on the device.

Essential Technologies for a Probing Attack

Front-side vs. back-side: Probing attack targets are those metal wires that carry assets, henceforth called target wires. The most common approach to reach target wires is to expose them from the back end of line (BEOL), that is, from the top metal layer towards silicon substrate. This is called a front-side probing attack.

A back-side probing attack, that is, probing that occurs through the silicon substrate. Back-side attack targets are not limited to wires. By exploiting a phenomenon during transistor activity, known as photon emission, transistors can also be probed to extract information.

Electrical probing vs. optical probing:

- Electrical probing is accessing an asset carrying signal via electrical connection.
- Optical probing techniques are often used in back-side probing to capture photon-emission phenomena during transistor switching. When transistors are switching, they spontaneously emit photons without external stimuli. By passively receiving and analyzing the photons emitted from a specific transistor, the signal processed by that transistor can be inferred.
- Compared to electrical probing, the optical approach has the advantage of being a purely passive observation, which makes it very difficult to detect.

- In addition to photon emission analysis, laser voltage technique (LVX), or electro-optical frequency modulation (EOFM), are also used during back-side attacks.

Essential Steps of a Probing Attack

Decapsulation: The first stage of most invasive physical attacks is to either partially or fully removes the chip package in order to expose the silicon die. Acid solutions, such as fuming nitric acid combined with acetone at 60°C, are often used to remove plastic packages. Decapsulation can also be done from the back-side of the chip by removing the copper plate mechanically, without chemical etching.

Reverse Engineering: Reverse engineering is the process of extracting design information from something, typically to reproduce it. In the case of probing, reverse engineering is used to understand how the chip works, which requires that the layout and netlist be extracted. By studying the netlist, the attacker can identify the assets. One-to-one correspondence between the netlist and layout can then determine the locations of target wires and buses.

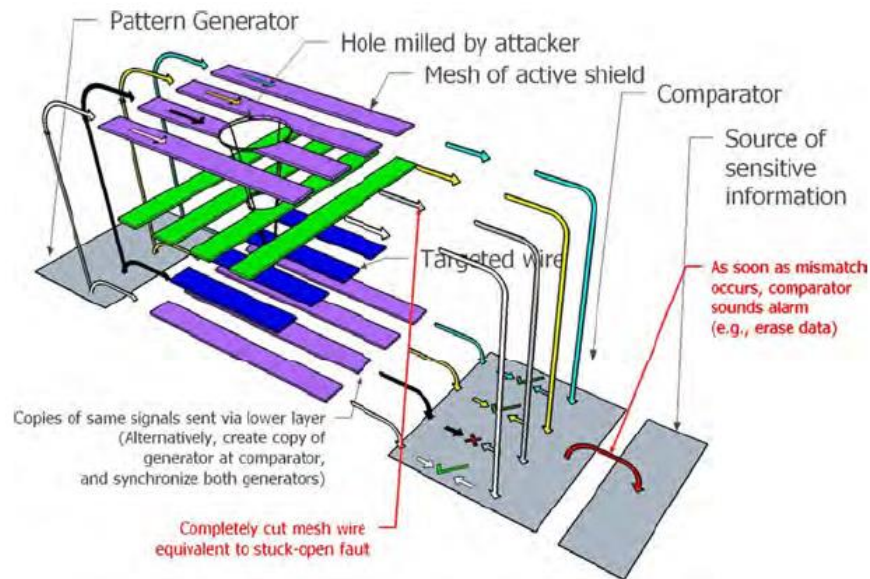
Locating target wires: Once the probing wire targets have been identified by reverse engineering, the next stage is locating the wires associated with the target on the IC under attack. The crux of the problem here is that while the attacker has located target wires on sacrificial devices during reverse engineering process, he/she now has to find the absolute coordinates of the point to mill blindly.

Reaching target wire and extracting information: With the help of modern circuit editing tools like FIB (Focused Ion Beam), a hole can be milled to expose the target wire. Once a target wire is exposed—assuming it is contacted without triggering any probing alarm signals from active or analog shields—the asset signals need to be extracted, for example, with a probe station.

EXISTING COUNTERMEASURES AND LIMITATIONS

A) Active Shields

- In this approach, a shield which carries signals is placed on the top-most metal layer to detect holes milled by FIB. The shield is referred to as “active” because signals on these top layer wires are constantly monitored to detect if milling has cut them.
- As shown in the figure, a digital pattern is generated from a pattern generator, transmitted through the shield wires on top-most metal layer, and then compared with a copy of itself transmitted from lower layer.
- If an attacker mills through the shield wires on top layer to reach target wire, the hole is expected to cut open one or more shield wires, thereby leading to a mismatch at the comparator and triggering an alarm signal to erase or stop generating sensitive information.
- Their biggest problems are that they impose large overheads on the design, but at the same time are very vulnerable to attacks with advanced FIBs, for example, circuit editing attacks.



B) Analog Shields and Sensors

An alternative approach to active shield is to construct an analog shield. Instead of generating, transmitting, and comparing digital patterns, analog shields monitor parametric disturbances with its mesh wires.

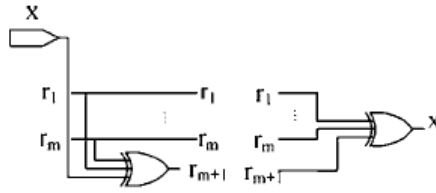
In addition to shield designs, the probe attempt detector (PAD) also uses capacitance measurement on selected security critical wires to detect additional capacitance introduced by a metal probe.

Compared to active shields, analog shields detect probing without test patterns and require less area overhead. The PAD technique is also unique in remaining effective against electrical probing from the back-side.

The problem with analog sensors or shields is that analog measurements are less reliable due to process variations, a problem further exacerbated by feature scaling.

C) t-Private Circuits

- In this technique, the circuit of a security-critical block is transformed so that at least $t + 1$ probe are required within one clock cycle to extract one bit of information.
- First, masking is applied to split computation into multiple separate variables, where an important binary signal x is encoded into $t + 1$ binary signals by XORing it with t independently generated random signals ($r_{t+1} = x \oplus r_1 \oplus \dots \oplus r_t$) as shown in Fig.
- Then, computations on x are performed in its encoded form in the transformed circuit. x can be recovered (decoded) by computing $x = r_1 \oplus \dots \oplus r_t \oplus r_{t+1}$. The major issue with t -private circuit is that the area overhead involved for the transformation is prohibitively expensive.



D) Other Countermeasure Designs

One known countermeasure that deters decapsulation stage of probing attacks is a light sensor that is sometimes included in a tamper-resistant design. Some other techniques include scrambling wires and avoiding repetitive patterns in shield mesh to impede the locating-target wire stage of probing attacks.

3. INVASIVE FAULT INJECTION ATTACK

- Invasive fault injection attack is realized through injecting faults by laser or focused ion beam (FIB) into a cryptographic device, and observing the corresponding outputs. Using differential fault analysis (DFA) methods, the secret key can be extracted.
- One example of optical fault injection techniques is a strong and precisely focused light beam that affects the behaviour of one or more logic gates in a circuit.
- A strong radiation of a transistor may form a temporary conductive channel in the dielectric, which, in turn, may cause the switch of a state. For example, by targeting one of the transistors in a static random-access memory (SRAM) cell, the value stored in this cell could be flipped up or down at will.
- Since the currents flowing inside a floating gate cell are much smaller than inside a SRAM cell, EPROM, EEPROM, and Flash memory cells are more vulnerable to fault injection attacks.
- Nowadays, common optical fault injection facilities consist of a laser emitter, focusing lens, and a placement surface with stepper motors to achieve an accurate focusing of the beam.
- Focused ion beam (FIB) is one of the most accurate and powerful fault injection technique that enables an attacker to edit the circuit, reconstruct missing buses, cut existing wires, and mill through layers by depositing or removing material on the circuit die.
- The countermeasures to prevent FIB-based fault injection attacks are almost the same with probing attacks.
- The basic strategies to prevent against fault injection attacks are intrusion detection, algorithmic resistance, and error detection.

Design for Security

SECURITY ARCHITECTURE

The typical approach for developing a baseline secure architecture depends on the following two steps:

- Use threat modeling to identify potential threats to the current architecture definition.
- Refine the architecture with mitigation strategies covering the threats identified.

Trusted execution environment (TEE) is developed to isolate between code and sensitive data at different points of the system execution. One of the most common TEE architectures is the trusted platform module (TPM), which is an international standard for a secure crypto processor. It is designed to secure the hardware by integrating cryptographic keys into devices. It covers methods to securely generate cryptographic keys and limit their use, random number generator requirements, and capabilities, such as remote attestation, and sealed storage.

In addition to TPM three TEE frameworks specifically developed for SoC designs are: Samsung KNOX, Intel Software Guard Extension (SGX), and ARM Trust Zone.

Samsung KNOX

This architecture is specifically targeted toward smartphones, and provides secure separation features to enable information partition between business and personal content to coexist on the same system. In particular, it permits hot swap between these two content worlds, for example without requiring system restart. The key ingredient of this technology is a separation kernel that implements the information isolation. This architecture permits several system-level services, including the following:

- Trusted boot, that is, preventing unauthorized OS and software from being loaded onto the device at startup.
- Trust-zone-based integrity measurement architecture (TIMA), which continually monitors kernel integrity.
- Security enhancement (SE) for Android, an enforcement mechanism providing protection of system/user data based on confidentiality and integrity requirements through separation.
- KNOX container, which offers a secure environment in which protected business applications can run with guaranteed information separation from the rest of the device.

ARM Trust Zone

- Trust Zone technology is a system-wide approach to provide security on high-performance computing platforms.
- The Trust Zone implementation relies on partitioning the SoC's hardware and software resources, so that they exist in two worlds: secure and nonsecure.
- The hardware supports access control and permissions for the handling of secure/nonsecure applications, and the interaction and communication among them.
- The software supports secure system calls and interrupts for secure runtime execution in a multitasking environment.
- These two aspects ensure that no secure world resources can be accessed by the nonsecure world components, except through secure channels, enabling an effective wall-of-security to be built between the two domains.

Intel SGX

- SGX is an architecture for providing a trusted execution environment provided by the underlying hardware to protect sensitive application and user programs or data against potentially malicious, or tampered operating systems.
- SGX permits applications to initiate secure enclaves or containers, which serve as so-called "islands of trust". It is implemented as a set of new CPU instructions

that can be used by applications to set aside such secure enclaves of code and data.

- This enables 1) applications to preserve the confidentiality and integrity of sensitive data without disrupting the ability of legitimate system software to manage the platform resources; and 2) end users to retain control of their platforms, applications, and services even in the presence of malicious system software.

SECURITY POLICY ENFORCER

This module is responsible for enforcing security policies that are imperative for ensuring security at the hardware level.

SIDE-CHANNEL RESISTANT DESIGN

Different countermeasure techniques have been proposed to counter the power and electromagnetic side-channel attack. These countermeasures can be broadly categorized as hiding mechanism and masking mechanism

Hiding Mechanism

Hiding mechanisms attempt to eliminate the relationship between the leaked information and the secret data, that is, make the leaked information uncorrelated to the secret data. A side-channel attack typically depends on the signal-to-noise ratio (SNR) which is defined as follows:

$$SNR = \frac{var(signal)}{var(noise)}.$$

Here, the signal refers to the leaked power signal, which is correlated to the secret data and exploited by the adversaries to perform the side-channel attack. The noise refers to the power signal, which has no correlation to the secret data. The hiding mechanisms decrease the SNR either by increasing the noise or by decreasing the signal to counter the side-channel attack. These mechanisms mainly utilize randomization and equalization techniques to decrease the SNR.

Randomization:

These techniques attempt to increase the noise of the circuit to reduce the SNR by constantly changing the execution order, or by generating noise directly. One possible approach for applying randomization is by injecting white noise on the channel.

Equalization:

These techniques attempt to decrease the leaked power signal, which is correlated to the secret data and reduce the SNR. The main idea here is to make equal power consumption for all operations related to processing secret data. The equalization technique can be realized by a specific type of logic style, which consumes a constant amount of power.

Masking Mechanism

These mechanisms attempt to randomize the intermediate values (function of the sensitive information) of a cryptographic operation to break the dependencies between these values and the power consumption.

Unlike the hiding mechanisms, masking mechanisms are applied at the algorithmic level and can be implemented by standard CMOS logic gates.

The main idea here is to conceal each intermediate value by a random mask that is different for every execution. It ensures that the sensitive data is masked with a random value, which eliminates dependencies between the intermediate values and the power consumption.

Masking technique can be implemented by Boolean secret sharing technique, which is discussed below

Let us consider that X and K denote two intermediate values associated with the plaintext and the sub-key of a cryptographic operation. Also, let us consider that another variable Z which is expressed as $Z = X \oplus K$.

Z variable is a function of the intermediate value K and any operation on Z variable could leak some information about K .

Boolean masking technique attempts to secure the operation of Z by randomly splitting into two shares $M0$ and $M1$, expressed by the following equation:

$$Z = M0 \oplus M1.$$

$M1$ is referred to as the mask, and $M0$ is referred to as the masked variable.

$M0$ is derived such that $M0 = Z \oplus M1$.

PREVENT TROJAN INSERTION

These techniques consist of preventive mechanisms that attempt to thwart hardware Trojan insertion by attackers.

Introduction to Blockchain Technology

- Blockchain can be described as a data structure that holds transactional records and while ensuring security, transparency, and decentralization.
- It is considered as a chain of records stored in the forms of blocks which are controlled by no single authority. A blockchain is a distributed ledger that is completely open to any and everyone on the network. Once an information is stored on a blockchain, it is extremely difficult to change or alter it.
- Each transaction on a blockchain is secured with a digital signature that proves its authenticity. Due to the use of encryption and digital signatures, the data stored on the blockchain is tamper-proof and cannot be changed.
- Blockchain technology allows all the network participants to reach an agreement, commonly known as consensus. All the data stored on a blockchain is recorded digitally and has a common history which is available for all the network participants. This way, the chances of any fraudulent activity or duplication of transactions is eliminated without the need of a third-party.

Working of Blockchain

A blockchain is a chain of blocks that contain data or information. Each block in a blockchain network stores some information along with the hash of its previous block. A hash is a unique mathematical code which belongs to a specific block. If the information inside the block is modified, the hash of the block will be subject to modification too. The connection of blocks through unique hash keys is what makes blockchain secure.

While transactions take place on a blockchain, there are nodes on the network that validate these transactions. In Bitcoin blockchain, these nodes are called as miners and they use the concept of proof-of-work in order to process and validate transactions on the network. In order for a transaction to be valid, each block must refer to the hash of

its preceding block. The transaction will take place only and only if the hash is correct. If a hacker tries to attack the network and change information of any specific block, the hash attached to the block will also get modified.

The breach will be detected as the modified hash will not match with the original one. This ensures that the blockchain is unalterable as if any change which is made to the chain of blocks will be reflected throughout the entire network and will be detected easily.

Types of Blockchains

There are two broad categories in which blockchains can be classified majorly i.e. Public and Private blockchains.

Public Blockchain- As the name suggests, a public blockchain is a permissionless ledger and can be accessed by any and everyone. Anyone with the access to the internet is eligible to download and access it. Moreover, one can also check the overall history of the blockchain along with making any transactions through it. Public blockchains usually reward their network participants for performing the mining process and maintaining the immutability of the ledger. An example of the public blockchain is the Bitcoin Blockchain.

Public blockchains allow the communities worldwide to exchange information openly and securely. However, an obvious disadvantage of this type of blockchain is that it can be compromised if the rules around it are not executed strictly. Moreover, the rules decided and applied initially have very little scope of modification in the later stages.

Private Blockchain- Contrary to the public blockchain, private blockchains are the ones which are shared only among the trusted participants. The overall control of the network is in the hands of the owners. Moreover, the rules of a private blockchain can be changed according to different levels of permissions, exposure, number of members, authorization etc.

Private blockchains can run independently or can be integrated with other blockchains too. These are usually used by enterprises and organizations. Therefore, the level of trust required amongst the participants is higher in private blockchains.

Features of Blockchain

Decentralised

Blockchains are decentralized in nature meaning that no single person or group holds the authority of the overall network. While everybody in the network has the copy of the distributed ledger with them, no one can modify it on his or her own. This unique feature of blockchain allows transparency and security while giving power to the users.

Peer-to-Peer Network

With the use of Blockchain, the interaction between two parties through a peer-to-peer model is easily accomplished without the requirement of any third party. Blockchain uses P2P protocol which allows all the network participants to hold an identical copy of transactions, enabling approval through a machine consensus. For example, if you wish to make any transaction from one part of the world to another, you can do that with blockchain all by yourself within a few seconds. Moreover, any interruptions or extra charges will not be deducted in the transfer.

Immutable

The immutability property of a blockchain refers to the fact that any data once written on the blockchain cannot be changed. To understand immutability, consider sending email as an example. Once you send an email to a bunch of people, you cannot take it back. In order to find a way around, you'll have to ask all the recipients to delete your email which is pretty tedious. This is how immutability works.

Once the data has been processed, it cannot be altered or changed. In case of the blockchain, if you try to change the data of one block, you'll have to change the entire blockchain following it as each block stores the hash of its preceding block. Change in one hash will lead to change in all the following hashes. It is extremely complicated for someone to change all the hashes as it requires a lot of computational power to do so. Hence, the data stored in a blockchain is non-susceptible to alterations or hacker attacks due to immutability.

Tamper-Proof

With the property of immutability embedded in blockchains, it becomes easier to detect tampering of any data. Blockchains are considered tamper-proof as any change in even one single block can be detected and addressed smoothly. There are two key ways of detecting tampering namely, hashes and blocks.

Each hash function associated with a block is unique. You can consider it like a fingerprint of a block. Any change in the data will lead to a change in the hash function. Since the hash function of one block is linked to next block, in order for a hacker to make any changes, he/she will have to change hashes of all the blocks after that block which is quite difficult to do.