CRESS: Framework for Vulnerability Assessment of Attack Scenarios in Hardware Reverse Engineering

Matthias Ludwig*†, Alexander Hepp†, Michaela Brunner†, Johanna Baehr†
*Infineon Technologies AG, Munich, Germany
matthias.ludwig@infineon.com
†Technical University of Munich, Department of Electrical and Computer Engineering, Munich, Germany
{alex.hepp, michaela.brunner, johanna.baehr}@tum.de

***Abstract*—Trust and security of microelectronic systems are a major driver for game-changing trends like autonomous driving or the internet of things. These trends are endangered by threats like soft- and hardware attacks or IP tampering – wherein often hardware reverse engineering (RE) is involved for efficient attack planning. The constant publication of new RE-related scenarios and countermeasures renders a profound rating of these extremely difficult. Researchers and practitioners have no tools or framework which aid a common, consistent classification of these scenarios. In this work, this rating framework is introduced: the common reverse engineering scoring system (CRESS). The framework allows a general classification of published settings and renders them comparable. We introduce three metrics: exploitability, impact, and a timestamp. For these metrics, attributes are defined which allow a granular assessment of RE on the one hand, and attack requirements, consequences, and potential remediation strategies on the other. The system is demonstrated in detail via five case studies and common implications are discussed. We anticipate CRESS to evaluate possible vulnerabilities and to safeguard targets more proactively.**

***Index Terms*—hardware reverse engineering (RE), security framework, vulnerability assessment, countermeasures, threat analysis, case study, IC trust**

I. Introduction

Hardware attacks and countermeasures have long been in an arms race. Increasing numbers of methods are being published on both ends, and there is no end in sight to this escalation. On the one hand, the extreme cost pressure of the electronics industry in general exacerbates the situation. Players in the hardware supply chain are forced to take measures that serve the profitability of their respective companies. Outsourcing at all levels – ranging from third party intellectual property (IP) cores, to outsourced testing – is at the top of the agenda. Due to these economic factors, trustworthiness and security often play only a secondary role. On the other hand, in complex systems, especially if safety- or security-critical applications are involved, the hardware is a favorable target. Once an attack on hardware is achieved, it may impact every single device in the field. In this case, sophisticated players will not shy away from chosen objectives, even if the expenditure is tremendous [1]. However, the focus is not only on *high-level* targets, so that "[...] the database of common vulnerabilities and exposures (CVEs) is just the tip of an iceberg" [2]. Attack scenarios are manifold and, albeit not considered an attack, hardware reverse engineering (RE) has proven its relevance in these [3]. Scenarios engaged with RE are at an all-time high. This trend is accompanied by an increasingly difficult assessment of the published scenarios. On both the defensive and offensive side, researchers and practitioners face difficulties to properly rate the consequences of published measures and potential countermeasures involving hardware RE. The hardware is the established root-of-trust, but one issue will remain: the closer a potential vulnerability is to the *physical* implementation, the lesser the chances are that it can be *updated* or patched. Covert side-channels, malicious players in the supply chain, or deficiently implemented security features all fall into this category. The offensive and defensive role of RE in all mentioned scenarios has been shown in the past, yet a common framework for an adequate assessment is still pending. It is this very gap that the Common Reverse Engineering Scoring System (CRESS) fills: Providing an evaluation framework for the comprehensive assessment of RE-involved exploitation scenarios. The contributions of this paper are:

- In Section II, state-of-the-art circuit RE methods are structured and their link to hardware attacks are discussed.
- In Section III, the CRESS is introduced. All metrics are discussed in detail and the multitude of attributes to be considered are outlined. All attributes and values are highlighted as Attr Value.
- In Section IV, the framework is thoroughly discussed via five selected case-studies: stealthy dopant Trojans [4], RE improved laser fault attacks [5], RISC-V cryptographic Trojans [6], and two studies that address the issue of IP piracy. All case-studies are rendered comparable through the framework and commonalities and distinctions are examined.

This work was partly funded by the German Ministry of Education and Research in the project RESEC under Grant No.: 16KIS1009. DOI: 10.1109/PAINE54418.2021.9707695

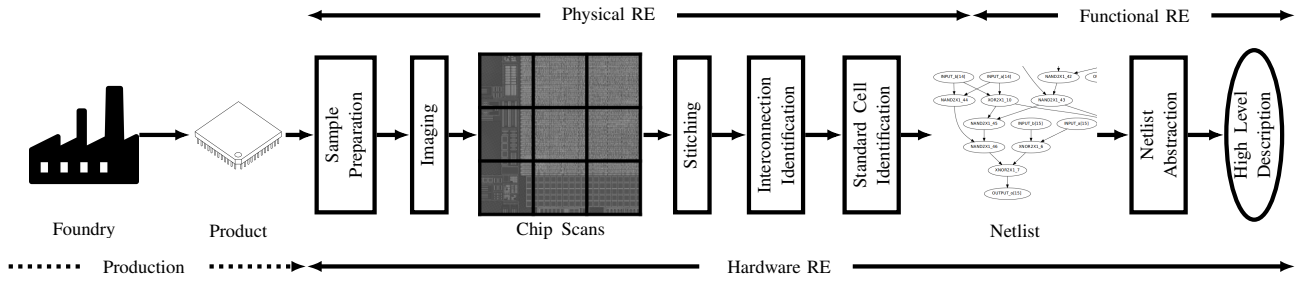

Figure 1. Process flow for hardware reverse engineering of integrated circuits.

II. Hardware Reverse Engineering

A. General RE Overview

In general, hardware RE activities can be distinguished into two categories: actions with benevolent intentions and actions with malicious intentions. The former are, for example, studies to understand a new technology or the prevention of patent infringements. Malicious activity is aimed at copying IP or in support of subsequent hardware-based attacks. Despite the different intents behind RE, the individual methodologies are alike. These can be classified into three main categories [7]:

- System level tear-downs: Identification of the product, package, internal boards, and components (printed circuit board (PCB) level)
- Technology analysis: Evaluation of the manufacturing process. Structures, materials, and geometrical dimensions can be examined (integrated circuit (IC) level)
- Layout and circuit extraction: Gradual delayering of a device including a layout extraction and a netlist and schematic reconstruction (IC or PCB level)

More precisely, layout and circuit extraction, or circuit RE, can be divided into a *physical* analysis part and a *functional* recovery part. These processes are briefly described in the following. The alphabetical coding of every sub-process indicates the intermediate *RE Results* RR. These values are important for the subsequent rating in the CRESS in section III-B.

B. Physical RE – Packaged IC to Netlist

The *physical* part of RE, as depicted in Figure 1, is a well-studied process [8]–[10] and can be executed in a sequential order.

Sample Preparation. The first RE step is the physical preparation of the packaged IC. This process step can be split into two sub-processes: In the *depackaging* step, the die is removed from its package and the bare silicon IC is exposed RR DC. Usually, a cross-section of the IC is produced RR CS. Optical measurement data is acquired through optical microscopes or scanning electron microscopes. This allows technology measurements such as the number of metallization layers, the process node, the type of the technology, or a material analysis RR TM. If a database with technological features exists, a one-to-one technology identification is possible RR TI. The technological information aids a *delayering* of the die. In the course of this, the layers of the device – ranging from the top metal down to the active areas – are individually prepared for imaging.

Imaging. The results after delayering are generated by imaging systems. Via an optical microscope, images of individual layers and a rudimentary determination of the IC's floorplan (dieshot) are possible RR DS. Larger building blocks can be isolated within the layout. For circuit reconstruction, microscopes with far better resolution are necessary. Scanning electron microscopes (SEMs) or specialized chip scanners are required for this task. Unstitched raw images from every layer form the intermediate RE result RR US after scanning.

Image Stitching. Stitching is the process to reconstruct two-dimensional, geometrically-undistorted mosaics RR SL. By using overlapping portions of adjacent images, an area-based feature extraction algorithm allows aligning images and creates a SEM image mosaic [11]. These mosaics are semi-automatically aligned to recreate a three-dimensional stack of superimposed layers RR LA.

Interconnection Identification. The reverse engineered layout is extracted from the mosaic by image processing. The result is a vector-representation of the layout data of the original integrated circuit, segmented into metal lines, vertical interconnection accesses, and contacts RR GD.

Standard Cell Identification. The repeatedly placed standard cells are classified through clustering algorithms utilizing inter-cell similarity scores or other feature extraction techniques. From these clusters, a standard cell library can be reverse engineered, from which the transistor netlist is obtained manually or which is merely described as boolean logic RR SI [10].

Netlist Re-Routing. To obtain a flat netlist, the wires between standard cells are identified by connecting overlapping conductive patterns (i.e. VIAs, wires, and contacts) [12]. The re-routed flat netlist is the final step of the physical RE process RR FN.

For a full circuit RE, all discussed steps are necessary to successfully recover a netlist. Yet, depending on the specific analysis target, intermediate results can be sufficient and the full RE of the netlist may not be necessary.

C. Functional RE – Netlist Abstraction

The *functional* part of RE [13]–[15], as depicted in Figure 1, extracts high-level information of the gate-level netlist. In contrast to the physical part of RE, the single steps of the functional part are executed sequentially or simultaneously,

depending on the targeted high-level information. Also, some steps can be performed with or without specific previous steps. This affects the complexity to derive results or even the possibility for a correct extraction of information. As a consequence, the following RR values are listed without intending a specific order.

Netlist Partitioning. A common starting point of functional RE is netlist partitioning RR PN. In this step, the flat netlist is split into smaller submodules or into a hierarchy of submodules. This is a divide-and-conquer approach to analyse all submodules separately. Various approaches for netlist partitioning exist, like for example separation along the data path by word identification [16], functional block identification [17] or graph partitioning [18].

Input/Output (I/O) Signal Identification. Reconstructing input and output signals RR HS in a given partial or complete netlist can aid further analyses [16]. An identification of I/O blocks increases abstraction and aids further analysis tasks like a differentiation between combinatorial and sequential paths. The aim is to obtain a "data-sheet" representation of a given module or submodule.

Data Path Identification. In order to gain more information about an unknown netlist, data words can be identified and the data path can be extracted RR DI. There are different approaches for data path identification, like for example structural-based and similarity-based approaches [19], [20], functional-based approaches [21], or a combination of both [22].

Functional Block Identification. One or multiple submodules of the netlist (after partitioning) can be matched to a library of known netlists or netlist structures to identify their high-level functionalities [16], [23]–[25]. Some approaches require a perfect match RR PF while other approaches enable fuzzy matching with only similar netlists but equal functionality RR PH.

Control Logic Identification. A special part of high-level netlist RE is the extraction of the design's control logic or finite state machine (FSM) RR CI. The identification of gates and wires which belong to the control logic usually starts with the identification of the FSM's flip-flops [26], [27] and continues with identification of the corresponding gates, wires and inputs or flip-flop outputs [26], [28]. Finally, the extracted netlist part is evaluated by all possible input combinations to reveal the corresponding FSM, assuming a specific reset state [29].

Complete Functional Identification. All or some of the previous steps can be combined to extract up to the complete functionality of a design RR CF [16]. One can differentiate between the identification of the complete overall functionality or the identification of the concrete implementation to achieve this complete overall functionality.

D. Attack Scenarios – A RE Perspective

The attack scenarios can be classified into direct and indirect attack vectors as shown in Figure 2. The direct attack vector includes all scenarios in which RE is used to recover the IP. It covers *IP confidentiality* *IC related topics, like counterfeiting, IP theft and re-use, or overproduction, but also *IP integrity* *II related topics, like IP manipulation or hardware Trojan insertion. In addition, a second, indirect attack vector can be identified. It includes all scenarios where RE has a supporting role to attack the product or in this case more precisely the device's application. A well-known example is the RE-assisted attack by Nohl *et al.* [3]. The specific attacks, including logical, observing, semi-invasive, or invasive, are improved or even made possible by prior RE results. Similar to direct attacks, indirect attacks can be classified to affect *on-device confidentiality* *OC, *on-device integrity* *OI and *on-device availability* *OA. Both attack vectors must be considered when assessing RE-improved attacks.

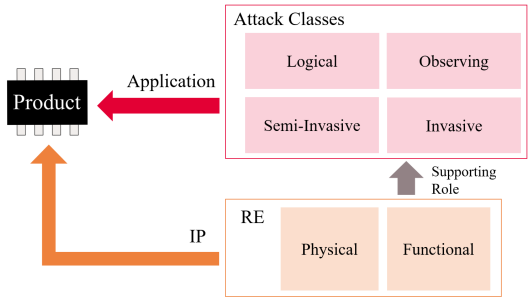

Figure 2. A generic *product* in the center of the IP and application vectors. Conventional attacks targeting applications can be supported by RE.

III. Scoring of RE Vulnerabilities

A. The Common Vulnerability Scoring System (CVSS)

For the quantitative and qualitative assessment of common vulnerabilities and exposures (CVEs), the common vulnerability scoring system (CVSS) [30] was introduced in 2006. Originally, it was set up by the National Infrastructure Advisory Council (NIAC) and is now maintained by the non-profit organization FIRST Inc. In 2015, the third version of the framework was published, which gradually improved. The CVSS provides a framework wherein users can evaluate the severity of software vulnerabilities. The goal is to provide open and universal standard severity ratings of vulnerabilities. The framework has emerged into an established measurement system for the industry, organizations, and governments that need accurate and consistent vulnerability severity scores.

The output of the system is a quantitative score and a vector string, in which the individual criteria are listed. The three areas of concern are: base metric, including exploit and impact scoring, temporal metric, and environmental metric. In summary, the CVSS provides a well-accepted mechanism to analyze vulnerable software components. The system is also used on hardware-specific attacks like Meltdown [31] or Rowhammer [32]. Yet, hardware RE and hardware RE-assisted attacks need different exploit, impact, but also temporal metrics to be able to address the multi-faceted field of RE comprehensively and to compare RE attacks reasonably.

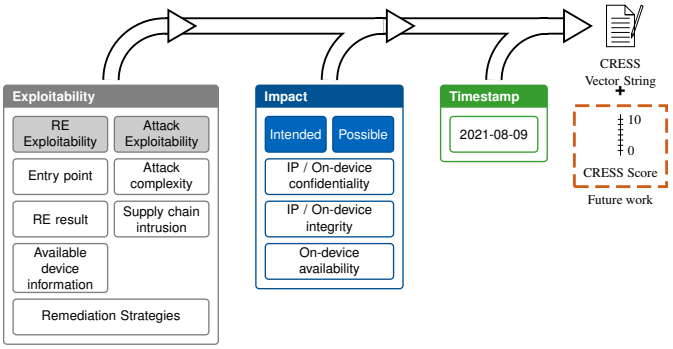

Figure 3. Overview of the CRESS. The metrics result in a unique vector string for an individual scenario.

B. The Common RE Scoring System (CRESS)

The introduced CRESS framework adapts the original scoring attributes of CVSS towards hardware RE-related specifics which are discussed in the following section. The primary division into base, temporal, and environmental metrics is not adopted, because for RE-related attacks the temporal attributes are much more related to the base attributes. The success of an attack using RE depends, for example, on the available remediation strategies, which again strongly improve over time. Therefore, temporal attributes should be included into the CRESS framework by default. On the other hand, the impact is treated as separate metric, because several scenarios can be distinguished, see section II-D. Summarizing, the following three CRESS metrics are introduced: exploitability, impact, and a timestamp. An overview of the CRESS is depicted in Figure 3.

1) Exploitability: The exploitability metric is divided into two categories: the exploitability of the necessary RE results for the rated attack, and the exploitability of the attack under the assumption that the necessary RE results are available.

a) RE Exploitability: For a rating of the RE exploitability, three attributes for a qualitative assessment are defined: entry point, RE result, and available device information.
The first attribute *entry point* $\boxed{\text{EP}}$ determines the role an actor has within the supply chain. When using CRESS, the most intuitive way to address single features is by asking a specific question: *Who is the attacker?* Different players possess different characteristics which must be rated distinctly. The end customer $\boxed{\text{E}}$ only has access to public data and has no rights within the supply chain other than to request information. All other entry points are directly involved within the supply chain and can potentially abuse their permissions maliciously. The foundry $\boxed{\text{F}}$ for example has access to the physical layout and process design kit (PDK) information. Further involved parties include backend designer $\boxed{\text{B}}$, the frontend designer, either in a partial position $\boxed{\text{P}}$ or in a leading position $\boxed{\text{L}}$, and third party IP providers $\boxed{\text{I}}$. The backend designer might have limited possibilities to change high-level functionalities undetected, but has access to the layout information. The frontend designers have access to the register-transfer level (RTL) descriptions, where a supporting role is less powerful than a leading role. Finally, the IP provider has the full control over its offered IP, but has no access to the remaining components of the design. In summary, the $\boxed{\text{EP}}$ set consists of following elements:

$$\boxed{\text{EP}} := \left\{\boxed{\text{E}}, \boxed{\text{F}}, \boxed{\text{B}}, \boxed{\text{P}}, \boxed{\text{I}}, \boxed{\text{L}}\right\}$$

The second attribute – *RE Results* $\boxed{\text{RR}}$ – has been introduced in section II-B. The question to be asked is: *Which RE result is necessary to arrive at the target to be exploited?* Users can choose from the predefined set of RE results. For improved usability, this category is sub-divided into physical RE results $\boxed{\text{RR}_{\text{phys.}}}$ and functional RE results $\boxed{\text{RR}_{\text{func.}}}$. Both sets are defined as follows:

$$\begin{aligned}
\boxed{\text{RR}} &:= \boxed{\text{RR}_{\text{phys.}}} \cup \boxed{\text{RR}_{\text{func.}}} \\
\boxed{\text{RR}_{\text{phys.}}} &:= \left\{\boxed{\text{DC}}, \boxed{\text{CS}}, \boxed{\text{TM}}, \boxed{\text{TI}}, \boxed{\text{DS}}, \boxed{\text{US}}, \boxed{\text{SL}}, \right. \\
&\qquad \left. \boxed{\text{LA}}, \boxed{\text{GD}}, \boxed{\text{SI}}, \boxed{\text{FN}}\right\} \\
\boxed{\text{RR}_{\text{func.}}} &:= \left\{\boxed{\text{PN}}, \boxed{\text{HS}}, \boxed{\text{PH}}, \boxed{\text{CI}}, \boxed{\text{DI}}, \boxed{\text{PF}}, \boxed{\text{CF}}\right\}
\end{aligned}$$

Section II already explains that the set $\boxed{\text{RR}_{\text{phys.}}}$ is sorted hierarchically, while the set $\boxed{\text{RR}_{\text{func.}}}$ is not, because there is no clear sequential order for functional RE. Consequently, for $\boxed{\text{RR}_{\text{phys.}}}$, the user selects the hierarchically highest value exclusively. For $\boxed{\text{RR}_{\text{func.}}}$, the user has to decide what is the most relevant result for the subsequent attack.

Third, the *available device information* $\boxed{\text{AI}}$ is rated. The question to be asked is: *Is the targeted design openly available?* A rough differentiation can be done into fully proprietary and open-source designs. Proprietary or closed-source $\boxed{\text{C}}$ designs offer only publicly available information to a potential attacker. Open-source designs can be distinguished into partial open-source designs $\boxed{\text{P}}$ and fully open-source designs $\boxed{\text{O}}$. For instance, RISC-V designs in which the crypto primitives remain proprietary can be ranked partly as open source. Fully open-source products incorporate open-source IP in all modules of a design. This may also include the standard cell library, analog, or memory IP. Thus, the $\boxed{\text{AI}}$ set consists of the following elements:

$$\boxed{\text{AI}} := \left\{\boxed{\text{C}}, \boxed{\text{P}}, \boxed{\text{O}}\right\}$$

The three attributes of the RE exploitability should result in an assessment of the RE complexity. There is a strong relation between the entry point and the RE result, because depending on the entry point, RE results are significantly more or less difficult to reach.

b) Attack Exploitability: For a rating of the attack exploitability, two attributes for a qualitative assessment are defined: attack complexity, and supply chain intrusion.

The first attribute *attack complexity* $\boxed{\text{AC}}$ rates the complexity of a RE-improved attack under the assumption that the necessary RE results are given. The question to be asked is: *How complex is the targeted attack based on a given RE*

result? To avoid an overly complex scoring, five attributes can be chosen to assess the attack complexity: Unavailable N, low L, medium M, high H, extreme X. The value N is necessary to cover pure RE scenarios wherein only the direct IP vector constitutes an attack. The low attack complexity represents hobby projects of individuals, medium attack complexity represents non-dedicated labs, high attack complexity represents labs dedicated for the attack purpose, and extreme attack complexity represents state actors with basically infinite resources. Thus, the AC set consists of the following elements:

$$\text{AC} := \{\text{N}, \text{L}, \text{M}, \text{H}, \text{X}\}$$

Furthermore, the *supply chain intrusion* SI for the intended scenario is addressed, again under the assumption that the necessary RE results are already available. Different attack scenarios include different permissions in the design and manufacturing process. The questions to be asked is: *What are the permissions necessary to execute the attack?* If a hypothetical attacker requires only access to the end product N, he or she does not need any write or read capabilities. The same value is used for attack vectors in which RE is used directly by an adversary. Read only L access is, for instance, the ability to read the design files during a single design phase. Read and write access are the most extreme form of supply chain intrusion. This feature is additionally distinguished between single-stage M and multi-stage H permissions. Single-stage permission implies, for instance, that read and write access is only required on the physical layout. In case of multi-stage permission, read and write accesses are required at several points in the production process. For example, a potential attacker may need read access to the synthesized netlist in order to subsequently modify the RTL description by writing to it. Thus, the SI set consists of the following elements:

$$\text{SI} := \{\text{N}, \text{L}, \text{M}, \text{H}\}$$

c) Remediation Strategies: Finally, possible *remediation strategies* RS are addressed. These are considered for the RE as well as the attack exploitability. The question to ask here is: *What countermeasures can prevent the attack as a whole?* Again, to avoid an overly complex scoring, all remediation strategies are abstracted to four attributes: unavailable N, pre-silicon measures A**, detection-based postsilicon measures *O*, and logic-based measures **L. One can choose between the value N and an arbitrary combination of the remaining values. Presilicon measures are active measures, like locking techniques [33], camouflaging [34], [35], physical unclonable functions [36], measures for hardware implementations, like a fault attack resistant AES implementation [37], or physical obfuscation techniques [38]. Presilicon measures aim to protect the product from the beginning of the product life cycle. In contrast to presilicon measures, postsilicon measures can also be applied years after products are in the field and at least mitigate potential vulnerabilities of already existing devices. Postsilicon measures are detection-based approaches, like hardware Trojan detection tools [39] or RE verification processes [40]. Summarizing, the RS set consists of the following elements:

$$\text{RS} := \{\text{N}, \text{A}, \text{O}, \text{L}, \text{AO}, \text{OL}, \text{AL}, \text{AOL}\}$$

2) Impact: The second metric is the impact *I* of RE-related attacks. It can either be direct or indirect, including *IC and *II related topics for direct and *OC, *OI and *OA related topics for indirect impacts, see more detailed explanations in section II-D. Moreover, the CRESS tool distinguishes between impacts which are related to the intended aim of the RE attack I* and impacts which also possibly occur but which are not targeted by the RE attack P*. For both of these impact categories, all direct and indirect impacts can be rated, resulting in the following elements:

$$I := \{\text{IIC}, \text{III}, \text{IOC}, \text{IOI}, \text{IOA}, \text{PIC}, \text{PII}, \text{POC}, \text{POI}, \text{POA}\}$$

For each of these impact attributes, a user can choose between three generic severity values *X*: no N, minor L, and major H impact, see the elements in the following:

$$X := \{\text{N}, \text{L}, \text{H}\}$$

3) Timestamp: In order to address the temporal dependency of the scoring result (e.g. available remediation strategies, etc.), the CRESS framework includes a third metric *timestamp* T. It has the form of a calendar date string in RFC3339 "full-date format" [41]. The timestamp shall be picked to represent the time of scoring and shall change upon possible re-scoring at a later time.

In a similar vein to the CVSS, a concise machine-readable exchange format for the scoring result is defined: the CRESS Vector String. The CRESS Vector String consists of the short identifiers of each scoring attribute and the short identifiers of the scored values. Identifiers and values are separated by a colon (:), the resulting pairs are concatenated in the order as they were introduced in this section with a forward slash as a separator (/). The resulting CRESS Vector String is suitable for storage and exchange between researchers. An exemplary template of the CRESS Vector String with the chosen attribute values t_1-t_{16} looks as follows:
EP:t_1/RR:t_2/AI:t_3/AC:t_4/SI:t_5/RS:t_6/IIC:t_7/III:t_8/IOC:t_9/
IOI:t_{10}/IOA:t_{11}/PIC:t_{12}/PII:t_{13}/POC:t_{14}/POI:t_{15}/POA:t_{16}/
T:yyyy-mm-dd

IV. Examplary Case Studies

In the following, we will demonstrate how various attacks on IP and / or applications can be modeled using selected examples. These should also serve as an aid for the classification of existing and future attacks. All discussed attacks are summarized in Figure 4. It is shown that the scenarios represent a variety of attack exploitability attributes. As the attacks have

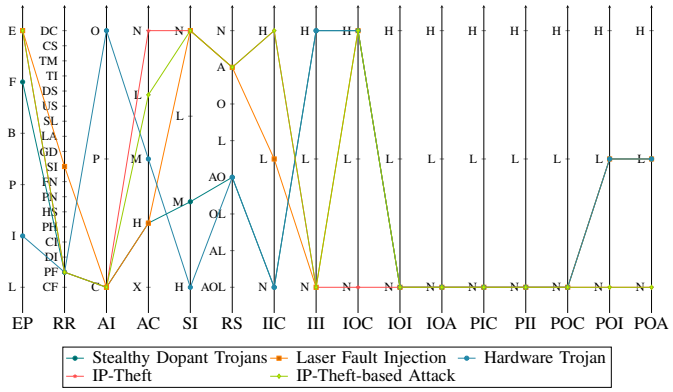

Figure 4. Overview of the CRESS rating of the presented case studies. The individual attribute values on the axes are ordered so that values with a high criticality are located at the top of the plot.

their limited and specific impact, only individual attributes are different from N. The discussed case-studies again highlight the importance of adequate remediation strategies for every single use-case.

A. Stealthy Dopant Trojans: Side-channel Hardware Trojan

The first evaluated case study are stealthy dopant Trojans by Becker *et al.* [4]. They introduced a covert type of hardware Trojan which only required modification of a doping mask. Since no optically observable changes were made to metal, VIA, or polysilicon masks, this type of hardware Trojan is not detectable through visible imaging methods. For the following assessment, we focus one of the two case studies: the side-channel leakage of parametrically modified *improved Masked Dual Rail Logic* cells. A modification of the p-doped region of an AND-OR-Inverter (AOI) gate reduced the effective width of individual transistors, and changed the dopant polarity of others and enabled a power side channel. Through this modification, the leaked key correlation is demonstrated via Correlation Power Analysis (CPA) on the SBox of an AES implementation.

RE The outlined attacker is a (potentially offshore) foundry EP F. Foundries manufacture commissioned ICs based on the provided physical design data. From the design data, a partial identification of the placed AOI gates of the targeted SBox implementation must be correctly located RR PF. The foundry does not require further information AI C.

Attack Expert knowledge is necessary to configure the dopant regions correctly without opening other side-channels or possibly manipulate the cell so that it is no longer functional AC H. Despite the high attack complexity, the supply chain intrusion is only medium SI M. A malicious actor requires read and write permissions merely to physical layout information.

Remediation strategies are also limited since the hardware Trojan was designed to be *stealthy*. Yet, Sugawara *et al.* [42] have shown that via destructive RE and a passive voltage contrast, for regions with changed dopant polarity, postsilicon detection strategies exist RS O. Viable presilicon remediation strategies RS A are for instance functionality obscuring via logic locking. Nonetheless, this method does not render the attack impossible for sophisticated attackers.

Impact The major on-device exploitability of the attack is the confidentiality exploitation through the side channel leakage IOC H and the IP integrity III H via the dopant area tampering. Minor possible compromises are the on-device integrity and availability POI L POA L.

CRESS Vector String EP:F/RR:PF/AI:C/AC:H/SI:M/RS:AO/IIC:N/III:H/IOC:H/IOI:N/IOA:N/PIC:N/PII:N/POC:N/POI:L/POA:L/T:2021-08-09.

B. RE Improved Laser Fault Injection

The second evaluated case study is RE improved laser fault injection by the example of an AES cipher by Courbon *et al.* [5]. For the attack, two devices which contain the targeted hardware implementation of an AES circuit are required. One of them is physically reverse engineered to obtain a stitched SEM image which is used to detect flip-flop pattern. The resulting coordinates are applied on the second device to identify promising fault injection positions. Together with additional knowledge about the AES circuit characteristics and timing behavior, the overall performance of the laser fault injection attack, which targets the extraction of the AES secret key, is improved.

RE The adversary EP E who has to physically reverse engineer one device to a stitched SEM image and must partially identify standard cells, namely flip-flops. Consequently, the required RE result is defined as the hierarchically highest result: standard cell identification RR SI. The adversary does not require further information AI C.

Attack The attacker needs to perform sophisticated fault injection and analysis, so the attack complexity can be rated high AC H. Additionally, no supply chain intrusion is required, because only two end products are necessary SI N.

The remediation strategy which can be assumed to be successful for this attack is a presilicon measure RS A. On the one hand, presilicon measures can increase the retrieval complexity of the necessary RE results, for example by camouflaging flip-flop pattern [43]. As a result, the coordinates of all potential flip-flop pattern positions have to be considered for the subsequent fault injections. On the other hand, an AES implementation can be used which is resistant to or strengthened against efficient and/or successful fault injection attacks [5], [37].

Impact The major on-device, intended impact of the attack is the confidentiality, because it targets the extraction of the secret AES key IOC H. Additionally, there is a minor intended impact concerning the IP confidentiality, because the SEM image and the flip-flop positions are revealed IIC L. The possible impact of the attack are minor impacts concerning the on-device integrity and availability caused accidentally by fault injections POI L, POA L.

CRESS Vector String EP:E/RR:SI/AI:C/AC:H/SI:N/RS:A/IIC:L/III:N/IOC:H/IOI:N/IOA:N/PIC:N/PII:N/POC:N/POI:L/POA:L/T:2021-08-09.

C. A RISC-V Cryptographic Chip with Hardware Trojans

The third evaluated case study investigates hardware Trojan insertion by the example of [6]. The authors taped-out an open-source microcontroller design with post-quantum cryptographic accelerators. This design was also infected by four hardware Trojans that threaten the security principles for this chip. In [6], the attacker is assumed to be a malicious IP provider of the microcontroller base design. The attacker therefore also provides the software compiler for this base design and can use this vector in combination with the hardware modification to enhance the capabilities of the hardware Trojans. In particular, one of the hardware Trojans allows leaking arbitrary data from the microcontroller through a covert channel in power usage or EM emanation.

RE The attacker is given to be the IP provider EP:I. To actually perform the attack of inserting a sophisticated Trojan as given in the paper, the attacker needs the information from a partial functional identification, in order to identify the source of leaked data, the trigger conditions etc. RR:PF. In the work, the design is described to be open source AI:O.

Attack The attacker needs to introduce additional logic only at the RTL level and into the compiler written in a high-level language, so the attack complexity can be rated medium AC:M. In order to perform the attack presented in the paper, read and write access to the supply chain is needed for the hardware design at the IP-provider and for the software compiler framework, which results in a high level of supply chain intrusion SI:H.

Against the hardware Trojans in this work, multiple remediation strategies are possible, such as hardware obfuscation to prevent the insertion, or hardware Trojan detection to observe the attack. Mere access control can not prevent the success of the Trojan attack RS:AO.

Impact The Trojan is a major modification of the design, impacting IP integrity III:H. The main purpose of the Trojan attack is to impact confidentiality by leaking arbitrary data on-device IOC:H. The possible impact of the attack are minor impacts concerning the on-device integrity and availability, occurring if the Trojan overheats the chip or causes voltage depletion due to the ring oscillator operation POI:L POA:L.

CRESS Vector String EP:I/RR:PF/AI:O/AC:M/SI:H/RS:AO/IIC:N/III:H/IOC:H/IOI:N/IOA:N/PIC:N/PII:N/POC:N/POI:L/POA:L/T:2021-08-09.

D. IC Piracy – IP Theft

The last two case studies focus on IP theft and the subsequent consequences. The first scenario considers the piracy of a specific design, for example the proprietary architecture of a multiplier. The design is reverse engineered and eventually reused in the attackers own design. The second scenario exploits the gained understanding of a proprietary cryptographic algorithm to detect a possible vulnerability.

RE Both operations are carried out by an end user EP:E, who requires at least the partial functional identification RR:PF of the respective module, in both cases to gain understanding of the design, to then either reuse the implementation or to attack the implementation. Both attacks are carried out on proprietary hardware designs AI:C.

1) Proprietary Multiplier Architecture:

Attack This scenario does not feature a subsequent attack, it already is the identification of the specific submodule AC:N SI:N. However, there exist remediation strategies to prevent the identification and reuse of the design. In particular, logic locking, logic camouflaging and watermarking are often used to prevent the piracy of proprietary hardware RS:A. Observation-based post-silicon methods do not exist, however reverse engineering competitors designs may allow insight into whether a design has been pirated.

Impact The impact of the scenario is N for all categories but the *IP confidentiality*. The theft of the proprietary design results in a IIC:H value.

CRESS Vector String EP:E/RR:PF/AI:C/AC:N/SI:N/RS:A/IIC:H/III:N/IOC:N/IOI:N/IOA:N/PIC:N/PII:N/POC:N/POI:N/POA:N/ T:2021-08-09.

2) Vulnerability Detection in a Proprietary Cryptographic Algorithm:

Attack The functionality of the cryptographic implementation must be understood to a point where critical design flaws can be understood for a subsequent attack. In this example we assume the easiest case: a weakness in the implemented cryptographic algorithm, for example using a SHA-1 hash function (broken in 2017). The attack may then be carried out by an individual with relatively inexpensive equipment AC:L. However, depending on the type of vulnerability, the attack complexity can range from low to high. For this IP-theft related scenario, no supply-chain intrusion is needed SI:N. The same remediation strategies apply to this scenario as to the previous scenario, plus, if applicable, the usage of a secured cryptographic implementation such as SHA-3 RS:A.

Impact As before, the attack directly impacts *IP confidentiality* IIC:H, as well as the confidentiality of the hashed data IOC:H. The on device integrity and availability remain unchanged, as no changes are made to the functionality of the chip.

CRESS Vector String EP:E/RR:PF/AI:C/AC:L/SI:N/RS:A/IIC:H/III:N/IOC:H/IOI:N/IOA:N/PIC:N/PII:N/POC:N/POI:N/POA:N/T:2021-08-09.

V. Conclusion & Outlook

The CRESS is a simply applicable, intuitive and easily extensible evaluation framework. Scenarios involving RE can be evaluated and future scientific discussions can be conducted far more efficiently with the help of this scheme. For the first time, it is possible to conduct analysis of hardware attacks that involve both the IP and application branches. Advances in RE, hardware attacks, and countermeasures can easily be integrated into the framework. This basic system already allows a coherent rating of all scenarios qualitatively. A web-based tool is available at the persistent uniform resource locator (PURL) `https://purl.org/cress/thetool`. Users can use the web-tool to score scenarios, to inspect given

CRESS vector strings, or to create graphs to compare multiple scenarios as in Figure 4.

Still, a quantitative assessment was intentionally not conducted in the set-up. A fair and coherent rating will be of extreme importance when the *CRESS Score* will be introduced. Consequently, to achieve a realistic numerical assessment, a similar strategy is targeted for the score as was done for the CVSS: the scoring of many RE-involed published scenarios by experts.

REFERENCES

[1] S. Adee, "The Hunt For The Kill Switch," *IEEE Spectr.*, 2008.

[2] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An Overview of Hardware Security and Trust: Threats, Countermeasures, and Design Tools," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2021.

[3] K. Nohl, D. Evans, S. Starbug, and H. Plötz, "Reverse-Engineering a Cryptographic RFID Tag," in *Proc. 17th Conf. Secur. Symp.*, 2008.

[4] G. Becker, F. Regazzoni, C. Paar, and W. Burleson, "Stealthy dopant-level hardware Trojans: Extended version," *J. Cryptogr. Eng.*, 2014.

[5] F. Courbon, P. Loubet-Moundi, J. J. A. Fournier, and A. Tria, "Increasing the efficiency of laser fault injections using fast gate level reverse engineering," in *2014 IEEE Int. Symp. Hardware-Oriented Secur. and Trust (HOST)*, 2014.

[6] A. Hepp and G. Sigl, "Tapeout of a RISC-V Crypto Chip with Hardware Trojans," in *Proc. 18th ACM Int. Conf. Computing Frontiers*. 2021.

[7] R. Torrance and D. James, "The State-of-the-art in Semiconductor Reverse Engineering," in *Proc. 48th Des. Automat. Conf.*, 2011.

[8] R. Quijada, R. Dura, and J. Pallares, "Large-Area Automated Layout Extraction Methodology for Full-IC Reverse Engineering," in *J. Hardware and Syst. Secur.*, 2018.

[9] B. Lippmann *et al.*, "Verification of physical designs using an integrated reverse engineering flow for nanoscale technologies," *Integration*, 2020.

[10] A. Kimura *et al.*, "A Decomposition Workflow for Integrated Circuit Verification and Validation," *J. Hardware and Syst. Secur.*, 2020.

[11] A. Singla, B. Lippmann, and H. Graeb, "Recovery of 2D and 3D Layout Informationthrough an Advanced Image Stitching Algorithm using Scanning Electron Microscope Images," *25th Int. Conf. Pattern Recognit.*, 2020.

[12] R. S. Rajarathnam, Y. Lin, Y. Jin, and D. Z. Pan, "ReGDS: A Reverse Engineering Framework from GDSII to Gate-level Netlist," 2020.

[13] L. Azriel, R. Ginosar, and A. Mendelson, "SoK: An Overview of Algorithmic Methods in IC Reverse Engineering," in *Proc. 3rd ACM Workshop Attacks and Solutions in Hardware Secur. Workshop*, 2019.

[14] M. Fyrbiak *et al.*, "Hardware reverse engineering: Overview and open challenges," in *2017 IEEE 2nd Int. Verification and Secur. Workshop (IVSW)*, 2017.

[15] L. Azriel, J. Speith, N. Albartus, R. Ginosar, A. Mendelson, and C. Paar, "A survey of algorithmic methods in IC reverse engineering," *Journal of Cryptographic Engineering*, pp. 2190–8516, 2021.

[16] P. Subramanyan *et al.*, "Reverse Engineering Digital Circuits Using Structural and Functional Analyses," *IEEE Trans. Emerg. Topics in Comput.*, 2014.

[17] J. Couch, E. Reilly, M. Schuyler, and B. Barrett, "Functional block identification in circuit design recovery," in *2016 IEEE Int. Symp. on Hardware Oriented Secur. and Trust (HOST)*, 2016.

[18] M. Werner, B. Lippmann, J. Baehr, and H. Gräb, "Reverse Engineering of Cryptographic Cores by Structural Interpretation Through Graph Analysis," in *3rd IEEE Int. Verification and Secur. Workshop*, 2018.

[19] N. Albartus, M. Hoffmann, S. Temme, L. Azriel, and C. Paar, "DANA Universal Dataflow Analysis for Gate-Level Netlist Reverse Engineering," *IACR Trans. on Cryptogr. Hardware and Embedded Syst.*, 2020.

[20] T. Meade, K. Shamsi, T. Le, J. Di, S. Zhang, and Y. Jin, "The Old Frontier of Reverse Engineering: Netlist Partitioning," *J. Hardware and Syst. Secur.*, 2018.

[21] C. Yu and M. J. Ciesielski, "Automatic word-level abstraction of datapath," in *IEEE Int. Symp. Circuits and Syst., ISCAS 2016*, 2016.

[22] W. Li *et al.*, "WordRev: Finding word-level structures in a sea of bit-level gates," in *IEEE Int. Symp. Hardware-Oriented Secur. and Trust*, 2013.

[23] W. Li, Z. Wasson, and S. A. Seshia, "Reverse engineering circuits using behavioral pattern mining," in *2012 IEEE Int. Symp. Hardware-Oriented Secur. and Trust*, 2012.

[24] J. Baehr, A. Bernardini, G. Sigl, and U. Schlichtmann, "Machine Learning and Structural Characteristics for Reverse Engineering," in *24th Asia and South Pacific Des. Automat. Conf. (ASPDAC'19)*, 2019.

[25] Y. Shi, B.-H. Gwee, Y. Ren, T. K. Phone, and C. W. Ting, "Extracting functional modules from flattened gate-level netlist," in *2012 Int. Symp. on Commun. and Inf. Technol. (ISCIT)*, 2012.

[26] K. S. McElvain, *Methods and apparatuses for automatic extraction of finite state machines*, US Patent No. US 6,182,268 B1, Filed Jan. 5th., 1998, Issued Jan. 30th., 2001, 2001.

[27] T. Meade, Y. Jin, M. Tehranipoor, and S. Zhang, "Gate-level netlist reverse engineering for hardware security: Control logic register identification," in *IEEE Int. Symp. on Circuits and Syst.*, 2016.

[28] Y. Shi, C. W. Ting, B.-H. Gwee, and Y. Ren, "A highly efficient method for extracting FSMs from flattened gate-level netlist," in *Proc. 2010 IEEE Int. Symp. Circuits and Syst.*, 2010.

[29] T. Meade, S. Zhang, and Y. Jin, "Netlist reverse engineering for high-level functionality reconstruction," in *2016 21st Asia and South Pacific Des. Automat. Conf. (ASPDAC'16)*, 2016.

[30] P. Mell, K. Scarfone, and S. Romanosky, "Common Vulnerability Scoring System," *IEEE Security Privacy*, 2006.

[31] M. Lipp *et al.*, "Meltdown: Reading kernel memory from user space," in *27th {USENIX} Sec. Symp.*, 2018.

[32] Y. Kim *et al.*, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014.

[33] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *J. Electron. Test.*, pp. 273–291, 2019.

[34] D. J. Forte, B. Shakya, H. Shen, and M. M. Tehranipoor, *Covert Gates To Protect Gate-Level Semiconductors*, US Patent No. US 2020/0273818 A1, Filed Feb. 21st., 2020, 2020.

[35] G. L. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "TimingCamouflage: Improving circuit security against counterfeiting by unconventional timing," in *Des., Automat. Test in Europe Conf. Exhibition*, 2018.

[36] T. McGrath, I. E. Bagci, Z. M. Wang, U. Roedig, and R. J. Young, "A PUF taxonomy," *Applied Physics Rev.*, 2019.

[37] H. Mestiri, N. Benhadjyoussef, and M. Machhout, "Fault Attacks Resistant AES Hardware Implementation," in *2019 IEEE Int. Conf. on Des. Test of Integr. Micro Nano-Syst. (DTS)*, Apr. 2019, pp. 1–6.

[38] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu, "Physical Design Obfuscation of Hardware: A Comprehensive Investigation of Device and Logic-Level Techniques," *IEEE Trans. Inf. Forensics Security*, 2017.

[39] A. Saini, G. Kundra, and S. Kalra, "A Survey on Hardware Trojan Detection: Alternatives to Destructive Reverse Engineering," in *Proc. 2nd Int. Conf on Comput., Commun., and Cyber-Secur.*, Singapore: Springer Singapore, 2021, pp. 885–897.

[40] "Enabling Trust for Advanced Semiconductor Solutions Based on Physical Layout Verification," in *Intell. Syst. Solutions for Auto Mobility and Beyond*, 2021.

[41] G. Klyne and C. Newman, "Date and Time on the Internet: Timestamps," Tech. Rep., 2002.

[42] T. Sugawara *et al.*, "Reversing Stealthy Dopant-Level Circuits," in *Cryptogr. Hardware and Embedded Syst.* 2014.

[43] M. Alam, S. Ghosh, and S. S. Hosur, "TOIC: Timing Obfuscated Integrated Circuits," in *Proc. 2019 Great Lakes Symp. VLSI*, 2019.