DISSERTATION

How Do Engineers Analyze Netlists? – Human Problem-Solving Processes in Hardware Reverse Engineering

Carina Yasmin Wiesen

Ruhr-Universität Bochum

Mai 2021

RUHR
UNIVERSITÄT
BOCHUM RUB

How Do Engineers Analyze Netlists? – Human Problem-Solving Processes in Hardware Reverse Engineering

DISSERTATION

zur Erlangung des Grades eines Doktor-Ingenieurs

der Fakultät für Elektrotechnik und Informationstechnik

an der Ruhr-Universität Bochum

vorgelegt von

Carina Yasmin Wiesen

geboren in Langenfeld (Rhld.)

Bochum, 31. Mai 2021

To my beloved family.

Erstgutachter:	**Prof. Dr.-Ing. Christof Paar**
Zweitgutachterin:	**Prof. Dr. Nikol Rummel**
Vorgelegt am:	31.05.2021
Verteidigt am:	21.09.2021

Abstract

Digital hardware systems are the foundation of our modern digital society with its innumerable interconnected digital devices, ranging from smartphones to the emerging Internet of Things (IoT) as well as traditional PCs and cloud servers. Digital hardware is realized in form of integrated circuits (ICs), i.e., microchips, that often perform various security-critical functions. They are, thus, attractive targets for attacks and malicious manipulations. This dissertation is concerned with a specific method to understand the inner structures and functionalities of microchips – hardware reverse engineering (HRE). On the one hand, HRE supports the analysis of ICs, for example, for the detection of malicious manipulations such as hardware Trojans or intellectual property (IP) theft. On the other hand, HRE can be applied to illegitimate ends that include, for example, product piracy, or the injection of malicious hardware backdoors and Trojans. Such malicious manipulations can have far-reaching consequences such as monetary losses, reputational damage for chip manufacturers, or security threats for critical infrastructures in which the manipulated chips are implemented. As tools that automate the entire HRE process do not yet exist, hardware reverse engineers have to make sense out of semi-automated HRE steps that are driven by human problem-solving processes and cognitive factors, such as intelligence or prior domain-specific knowledge. Consequently, the success of HRE strongly depends on the analysts' cognitive processes. However, the understanding of the underlying cognitive processes and factors in HRE have thus far not gained much attention in the research community, and remain largely unexplored and opaque.

In this doctoral thesis, I aim to analyze these poorly understood human problem-solving processes in HRE. An underlying motivation is to focus on the malicious aspects of HRE (e.g., IP theft; insertion of backdoors), as they are associated with severe consequences, and to prevent analysts from performing HRE with malicious goals. For this, it will be essential to provide ICs with appropriate protection mechanisms (i.e., obfuscation). We argue that a comprehensive understanding of not only the technical but especially the unexplored *cognitive processes* in HRE can help to derive ideas for future developments of appropriate obfuscation methods impeding HRE – which we coin cognitive obfuscation.

In the first part of this thesis, we present a methodological approach in order to circumvent the methodological problem that HRE experts are rare and thus, not widely available. In this context, we derive guidelines from educational and psychological research on skill acquisition to develop a university-level HRE course. Within two studies, we show that our HRE course promotes HRE skill acquisition in students with backgrounds in cyber security and thus, enable them to participate in our main study.

The second part forms the core of the dissertation. In this, we conduct our main study to systematically investigate human problem solving of hardware reverse engineers on different levels of expertise (i.e., eight intermediates; one HRE expert). All participants are asked to analyze a realistic gate-level netlist of an unknown IC. In order to investigate applied problem-solving strategies and their efficiency, we apply an iterative open coding scheme to analyze 2.445 single HRE actions of the nine participants. Our results reveal, for example, that two intermediates with above-average test results in the intelligence sub-factor working memory complete the HRE task in a solution time that is comparable to the solution time of the HRE expert.

In the final part of this thesis, I discuss our results on HRE problem-solving processes in the light of established psychological theories and taxonomies of human problem solving. Based on our findings, I conclude, for example, that HRE problem-solving performance may be a function of both expertise and cognitive abilities, that is also relevant in the context of cognitive obfuscation. In the context of our results, I develop initial ideas for cognitively challenging tasks that may raise the cost of HRE to an unattractive level, and therefore, may serve as an impulse for the development of cognitive obfuscation.

Kurzfassung (German Summary)

Digitale Hardwaresysteme sind die Grundlage unserer modernen digitalen Gesellschaft mit ihren unzähligen und miteinander verbundenen digitalen Geräten, die von Smartphones über das sich entwickelnde Internet der Dinge (IoT) bis hin zu traditionellen PCs und Cloud-Servern reichen. Digitale Hardware wird in Form von integrierten Schaltkreisen (ICs), d. h. Mikrochips, realisiert, die oft verschiedene sicherheitskritische Funktionen ausführen. Daher sind sie attraktive Ziele für Angriffe und bösartige Manipulationen. Diese Dissertation beschäftigt sich mit einer speziellen Methode, um die inneren Strukturen und Funktionalitäten von Mikrochips zu verstehen – dem Hardware Reverse Engineering (HRE). HRE unterstützt einerseits die Analyse von ICs, z.B. zur Erkennung von bösartigen Manipulationen wie Hardware-Trojanern oder von Diebstahl geistigen Eigentums (IP). Andererseits kann HRE auch zu illegitimen Zwecken eingesetzt werden, z. B. zur Produktpiraterie oder zum Einschleusen von bösartigen Hardware-Backdoors und Trojanern. Solche böswilligen Manipulationen können weitreichende Folgen haben, wie etwa monetäre Verluste, Reputationsschäden für Chip-Hersteller oder Sicherheitsbedrohungen für kritische Infrastrukturen, in denen die manipulierten Chips eingesetzt werden.

Aktuell gibt es kein Werkzeug, das den gesamten HRE-Prozess automatisiert, wodurch AnalystInnen gezwungen sind halbautomatisierten HRE-Schritten einen Sinn zu verleihen. Dies ist abhängig von menschlichen Problemlöseprozessen und kognitiven Faktoren wie Intelligenz oder domänenspezifischem Vorwissen. Demnach hängt der Erfolg von HRE stark von den kognitiven Prozessen der AnalystInnen ab. Das Verständnis der zugrundeliegenden kognitiven Prozesse und Faktoren in HRE hat jedoch bisher in der Forschungsgemeinschaft nicht viel Aufmerksamkeit erhalten und bleibt weitgehend unerforscht und undurchsichtig.

In dieser Dissertation ziele ich darauf ab, diese kaum verstandenen menschlichen Problemlöseprozesse in HRE zu analysieren. Eine zugrundeliegende Motivation ist es, sich auf die bösartigen Aspekte von HRE (z.B. IP-Diebstahl; Einfügen von Backdoors) zu konzentrieren, da diese mit schwerwiegenden Konsequenzen verbunden sind. Zudem sollen AnalystInnen perspektivisch davon abgehalten werden, HRE mit bösartigen Zielen durchzuführen. Dazu wird es unerlässlich sein, ICs mit geeigneten Schutzmechanismen (z. B. Obfuskation) auszustatten. Um HRE zu erschweren, argumentieren wir, dass ein umfassendes Verständnis nicht nur der technischen, sondern vor allem der unerforschten *kognitiven Prozesse* bei HRE helfen kann, Ideen für zukünftige Entwicklungen geeigneter Obfuskationsmethoden abzuleiten – die wir als kognitive Obfuskation bezeichnen.

Im ersten Teil dieser Arbeit stellen wir einen methodischen Ansatz vor, um das methodische Problem zu umgehen, dass HRE-ExpertInnen rar und daher nicht allgemein verfügbar sind. In diesem Zusammenhang leiten wir Richtlinien aus der pädagogischen und psychologischen Forschung zum Erwerb von Fähigkeiten ab, um einen HRE-Kurs auf Universitätsniveau zu entwickeln. Im Rahmen von zwei Studien zeigen wir, dass unser HRE-Kurs den Erwerb von HRE-Fähigkeiten bei Studierenden mit einschlägigem Hintergrund im Bereich der IT-Sicherheit fördert und sie somit befähigt, an unserer Hauptstudie teilzunehmen.

Der zweite Teil bildet den Kern der Dissertation. In diesem führen wir unsere Hauptstudie durch, um das menschliche Problemlösen von AnalystInnen (acht Nicht-ExpertInnen; ein HRE-Experte) systematisch zu untersuchen. Alle StudienteilnehmerInnen werden gebeten, eine realistische Netzliste eines unbekannten ICs zu analysieren. Um die angewandten Problemlösestrategien und deren Effizienz basierend auf 2.445 einzelne HRE-Aktionen der neun TeilnehmerInnen zu untersuchen, wenden wir ein iteratives offenes Kodierungsschema an. Unsere Ergebnisse zeigen beispielsweise, dass zwei Nicht-ExpertInnen mit einem überdurchschnittlich hohe Testergebnis im Intelligenz-Subfaktor Arbeitsgedächtnis die HRE-Aufgabe in einer Lösungszeit erledigen, die mit der Lösungszeit des HRE-Experten vergleichbar ist.

Im letzten Teil dieser Arbeit diskutiere ich unsere Ergebnisse zu HRE-Problemlöseprozessen im Licht etablierter psychologischer Theorien und Taxonomien des menschlichen Problemlösens. Aus unseren Ergebnissen schließe ich beispielsweise, dass die HRE-Problemlöseleistung eine Funktion sowohl der Expertise als auch der kognitiven Fähigkeiten sein kann, was wiederrum auch im Kontext der kognitiven Obfuskation relevant ist. Im Zusammenhang mit unseren Ergebnissen entwickle ich erste Ideen für kognitiv herausfordernde Aufgaben, die die Kosten von HRE auf ein unattraktives Maß erhöhen und somit als Impuls für die Entwicklung von kognitiver Obfuskation dienen können.

Acknowledgements

I would like to express my deepest appreciation to my both advisors, Nikol Rummel and Christof Paar, who have extraordinarily supported me throughout all these years of my PhD. My PhD was one of the most exciting journeys of my life and provided me with so many new impressions and experiences. I would like to thank Nikol Rummel very much for being such a great PhD advisor over all these years and always supporting me – whether it was writing papers, this thesis, or preparing for presentations at conferences and workshops. Furthermore, I would like to thank Christof Paar very much for his academic support, and for always encouraging and believing in me. Thank you for letting me become part of the Emsec family.

I would particularly like to thank my *Tandempartner* Steffen Becker. Without you, working on this topic would not have been possible – thank you for enabling this great and exciting interdisciplinary collaboration! I also look back with pleasure on our joint work trips - even if things didn't always go smoothly due to the weather and we ended up exploring the Boston airport, these were still great trips.

In this context, I am also extremely grateful for the interdisciplinary work environment provided by the graduate school SecHuman! I would like thank Susanne Kerstens, Anne Thiele, and Astrid Wichmann very much for their support in SecHuman.

I would like to take this opportunity to thank Sebastian Strauß for the great academic discussions and enriching exchanges about music during our shared office time. Thanks also for your great support in writing this thesis. It’s closing time! Furthermore, I would like to thank Leonie Schaewitz for being such a great office mate and for her support in writing my papers.

I am very thankful to Marc Fyrbiak and Nils Albartus for being so patient in explaining hardware reverse engineering to me. Nils, I will always remember our trips to Amherst and our Chaos-WG – Bacon everywhere! Furthermore, I would like to thank Max Hoffmann very much for his enormous support in programming the HRE game. I would like to thank Philipp Koppe for forcing me to climb a wall and thereby giving me my new favorite hobby, bouldering. I also hope that with you and Maik Ender many more bouldering and hiking adventures will follow.

I am also grateful for Anna Keune and Charleen Brand for their time correcting my "Denglish" – I really appreciate it! I am also thankful for the support of my PP-colleagues Julia Eberle, Julia Erdmann, Christian Hartmann, Valentina Nachtigall, Meike Osinski, and Selina Yek. Furthermore, I am thankful for the support of my colleagues from the Emsec-squad Susanne

Engels, Carl-Daniel Hailfinger, Endres Puschner, Falk Schellenberg, Julian Speith, Paul Staat, Johannes Tobisch, Sebastian Wallat, and Pascal Zimmer.

Many studies would not have been possible without the great support of the student assistants Isabell Bohm, Luisa Jansen, Zehra Karadag and René Walendy. Many thanks for everything!

I would also like to thank Nesrin Denizer, Adnan El Arja, and Irmgard Kühn very much for their support in organizing everything around my PhD.

Finally, but by no means least, I would like to thank my wonderful family Bernd Wiesen, Uta Wiesen, Vanessa Wiesen, and Marcus Haffmanns for their fantastic love and unbelievable support. Thank you so much for always believing in me and standing by my side!

Table of Contents

1 Introduction

The digitalization of our society is advancing at an ever-faster pace, making computing systems and communication technologies an integral part of our everyday lives. For example, the market for Internet of Things devices (IoT-devices) is constantly rising and is forecasted to reach $1.4 trillion by 2027 (Crane, 2021), connecting billions of users worldwide in the business, private, or automotive sectors. The digital hardware devices, usually given as integrated circuits (ICs) or microchips, are considered trusted in virtually any computing system. Although a variety of software components implement security mechanisms, they all depend on the underlying hardware devices for actually executing the security functions. Thus, the correct function of microchips is crucial for the security (and often also safety) of many if not most digital devices on which the digital society relies.

A commonly applied method to understand an IC's structure and functionalities is called Hardware Reverse Engineering (HRE). HRE is used to achieve legitimate goals (e. g., hardware Trojan detection, identification of counterfeit products, analysis of a competitor's product), but also to achieve illegitimate goals (e.g., malicious manipulations, overproduction and counterfeiting; intellectual property (IP) piracy) (Quadir et al., 2016). In terms of illegitimate goals, HRE can be applied during the global fabrication process of hardware chips, during which several (untrusted) stakeholders have access to valuable hardware designs and may be able to commit IP piracy or insert manipulations (Rostami, Koushanfar, & Karri, 2014). Those malicious manipulations in hardware chips are major concerns for many stakeholders, because the deployment of manipulated hardware chips in critical infrastructure (e.g., cellular networks; power grids; sensitive applications in aerospace or military) may have serious consequences and can compromise the security of an entire system (Guin et al., 2014; Guin, DiMase, & Tehranipoor, 2014; Becker, Regazzoni, Paar, & Burleson, 2013).

A gate-level netlist is defined as a logical circuit that is composed of gates and their interconnections (Becker et al., 2020). During HRE, high-level information is extracted from a low-level circuit during two major stages (Azriel, Ginosar, & Mendelson, 2019). First, the reverse engineer retrieves a gate-level netlist from the IC through several sub-steps including, for example, delayering and imaging of the single IC layers or post-processing of all images to assemble the gate-level netlist (overviews by Torrance & James, 2009 or Quadir et al., 2016). Besides this, it is also possible to obtain a gate-level netlist through online interception of design information. In a second step, a hardware reverse engineer aims to obtain a better understanding (i.e., sense-making) of the recovered netlist, the chip structure, or specific functionalities, for example, the identification of IP blocks (Becker et al., 2020). These processes of understanding the netlist are included in the second step of HRE (Azriel, Ginosar, & Mendelson, 2019) that have been so far only rudimentarily addressed in prior research.

In order to understand the netlist, hardware reverse engineers apply a set of semi-automated and customized tools, as fully automated HRE tools do not exist so far (Becker et al., 2020). Analysts typically apply HRE tools that enable the analysis of the present netlist, for example, by developing specific algorithms on the netlist (Chisholm, Eckmann, Lain, & Veroff, 1999), or by conducting manual and visual analysis of specific netlist components (Wallat et al., 2019). Due to the lack of tools that allow for fully-automated netlist analysis, HRE always involves the analysts' cognitive processes such as problem solving that is governed by their cognitive capabilities. Surprisingly, these cognitive processes, which determine the success of HRE, have been little explored in prior research and remain poorly understood.

Although there has been prior works that explored sense-making processes of software reverse engineers (Votipka, Rabin, Micinski, Foster, & Marzurek, 2020), or the effectiveness of software obfuscation methods (e.g., Ceccato et al., 2009; Ceccato et al., 2014), only little prior research has already dealt with the exploration of human factors in HRE. In terms of HRE, there is one relevant prior work by Lee and Johnson-Laird (2013) who explored problem-solving processes during reverse engineering of Boolean systems. The participants were asked to draw electrical circuits that controlled electrical lights or a water flow system. The authors conducted five laboratory experiments with students who had no prior domain-specific knowledge and showed that the difficulty of reverse engineering of Boolean systems is influenced by the number of components of the system and the dependencies of the components (Lee & Johnson-Laird, 2013). Furthermore, they found that students applied one of the two problem-solving strategies of either focusing on the output or single components. Based on their results, Lee and Johnson-Laird (2013) defined reverse engineering of Boolean systems as a specific type of human problem solving that has not been investigated by psychologists before. Since Lee and Johnson-Laird's (2013) study task lack in comparability to a realistic HRE tasks, it is questionable to what extent Lee and Johnson-Laird's (2013) results are applicable to problem-solving processes in HRE. Previously, we (Becker et al., 2020) conducted an exploratory investigation into the technical processes of HRE. We proposed a three-phased model of HRE consisting of i) candidate identification; ii) candidate verification; and iii) realization. We suggested that hardware reverse engineers have to pass through these three phases to successfully analyze a gate-level netlist. Furthermore, an initial comparison between the HRE processes of two reverse engineers showed that the faster participant chose a strategy comparable to divide-and-conquer, whereas the slower participant solved the HRE task through trial-and-error.

Within this doctoral thesis, I aim to achieve an overarching research goal: understanding human factors that play a role in HRE. The understanding of human factors in HRE is essential in order to estimate the effort of hardware reverse engineers or to support the development of sound countermeasures impeding HRE – suggested as cognitive obfuscation (Wiesen et al., 2019a; Becker et al., 2020). Although prior research has advanced the understanding of cognitive processes in reverse engineering (Lee & Johnson-Laird, 2013; Becker et al., 2020), it is necessary to conduct further explorations for the following reasons. Since Lee and Johnson-Laird (2013) conducted their studies under artificial laboratory conditions and included tasks that were limited in their comparability to realistic HRE settings, it is unclear if their results of cognitive processes in reverse engineering of Boolean systems are applicable to HRE problem solving. If we aim to achieve a comprehensive understanding of HRE problem solving and to derive ideas for future developments of cognitive obfuscation, it is necessary to conduct a study with realistic HRE task under realistic HRE conditions (e.g., including HRE tools).

However, researchers who aim to explore human factors in HRE face the methodological challenge that HRE experts often work for highly-specialized companies or government agencies which have intrinsic motivations **not** to reveal information about their HRE working processes. Furthermore, the worldwide population of HRE experts is also very small. Both aspects make the efforts to locate and recruit those experts not only very time-consuming, but also often unsuccessful.

Because human factors are strongly involved in HRE, researchers who aim to receive a better understanding of HRE need to apply an interdisciplinary approach. By combining hardware security and cognitive psychology, we tried to address this research problem. The realization of the interdisciplinary approach and research project was enabled by the interdisciplinary environment of the SecHuman graduate school. In the SecHuman graduate school, interdisciplinary tandem teams consisting of two doctoral students from different disciplines conduct research on an overarching security-related research question. Accordingly, most of the results and work steps originate from the interdisciplinary collaboration with my tandem partner Steffen Becker. My role in this tandem constellation was to explore human factors in HRE (i.e., problem-solving processes; the role of cognitive abilities and expertise in HRE) from a cognitive psychological perspective, whereas Steffen Becker's research focus was on the technical aspects of HRE.

Main Contributions of Doctoral Thesis

Cybersecurity topics in human-computer interaction (HCI) research have gained importance over the last decade. According to (Stephanidis et al., 2019), research on security has become one of the seven grand challenges for the HCI community. In this dissertation, methods and theoretical concepts of cognitive psychology and HCI research are adapted to explore the behavior of hardware reverse engineers who were asked to solve a realistic HRE task. Prior research has long considered the human factor in designing secure and usable systems with the goal to improve the interaction between security aspects and specific user groups, such as end users (e.g., Sasse, Brostoff, & Weirich, 2001) or software developers (e.g., Acar, Stransky, Wermke, Mazurek, & Fahl, 2017). However, the long-term goal of this doctoral thesis is not to improve the interaction between users (in our case hardware reverse engineers) and the system (a gate-level netlist as the HRE target), but rather to make HRE more difficult by deriving suggestions for sound countermeasures impeding HRE. This twist on the typical HCI framework (i.e., improving the interaction between user and system) may be a novel direction for the HCI community.

The development of initial ideas for cognitive obfuscation in the end of this dissertation is based on a multi-step research approach to receive a better understanding of human factors in HRE. The following list summarizes the main contributions of this dissertation:

1. **Embedding of technical HRE problems in psychological theory**

   Following the theoretical considerations by Lee and Johnson-Laird (2013), I defined underlying cognitive processes in HRE as problem-solving processes. The theoretical embedding in existing psychological theories of human problem solving builds the foundation to investigate cognitive processes and potentially influencing factors such as prior knowledge or intelligence in HRE in a theory-based and structured way.

2. **Development and evaluation of a methodological approach to explore human factors in HRE**

   As mentioned above, the recruitment of HRE experts is very costly and often unsuccessful. Since there is hardly any research on the cognitive processes in HRE, there are also no methodological recommendations on how to circumvent the problem that HRE experts are unavailable for research. Together with the interdisciplinary research team, I developed and evaluated a methodology that enabled students to acquire a sufficient amount of HRE skills and prepared those students to participate in the main study on exploring human factors in HRE.

3. Conducting of an empirical study to explore human factors in HRE

In collaboration with my co-authors, I designed and conducted a main study in order to explore underlying human factors in HRE. In this study, participants with different levels of HRE expertise were asked to solve a realistic HRE task. The participants' problem solving was explored by an iterative open coding procedure of 2445 single log entries. Besides behavioral log files, cognitive abilities and the level of prior domain-specific knowledge was measured in the main study.

4. Exploration of human factors in HRE

- **Exploration of problem-solving processes in HRE:** The overarching goal of this dissertation is to obtain a meaningful and comprehensive understanding of human factors in HRE, defined as problem-solving processes – processes that have been poorly investigated by prior research so far. Within the doctoral thesis, we explored the problem-solving performance of hardware reverse engineers who were asked to solve a realistic HRE task. Therefore, we conducted a main study on HRE problem solving and analyzed the problem-solving processes by applying an iterative open coding scheme based on the well-established Grounded Theory (Strauss & Corbin, 1998). Our results showed, for example, that it seemed as if a single best HRE strategy did not exist for the problem at hand and different problem-solving processes may lead to efficient solutions. In addition, we identified several difficulties that hardware reverse engineers faced during the HRE problem solving. The findings are also discussed in a broader context of established psychological literature on problem solving.
- **Exploration of cognitive factors in HRE problem solving:** Besides the exploration of problem-solving processes in HRE, we investigated whether the level of expertise or of cognitive abilities play a role in HRE problem solving. Our exploration showed that two intermediates solved the HRE task with an efficiency comparable to the HRE expert. Furthermore, our data showed that participants with above-average scores in the intelligence sub-factor working

memory tended to solve the HRE task faster than participants with lower working memory scores. Based on our results, I concluded that HRE problem solving may be a function of both expertise and cognitive abilities, and discuss this point in the light of psychological literature.

5. Deriving initial ideas for cognitive obfuscation

Beyond the contribution to the main goal (i.e., understanding human factors in HRE), I developed initial ideas for sound countermeasures impeding HRE – so called cognitive obfuscation (Becker et al., 2020; Wiesen et al. 2019) (see Chapter 6). I suggested that countermeasures impeding the cognitive processes of HRE should consider both expertise and cognitive abilities of analysts, as both factors may support efficient HRE problem solving. For example, prior psychological research suggested potential characteristics of tasks that have posed huge challenges for experts (e.g., Chi 2006). Against this background, I suggested to develop obfuscation tasks in which experts could not rely on their prior domain-specific knowledge and cannot apply standard-strategies through which these experts may lose efficiency in their HRE problem solving. As the cognitive factor working memory may also contribute to efficiently solving HRE tasks, I derived initial ideas for obfuscation tasks that aim to overload the capacity of hardware reverse engineers' working memory.

2 Theoretical Background and Research Structure

In Sections 2.1-2.5, I present relevant technical and psychological background in the context of HRE, and describe the methodological problem that HRE experts are unavailable for research. In Section 2.6, I build upon the presented background to present my research goals to analyze the so far poorly understood problem-solving processes in HRE. Finally, I provide an overview of the structure of the dissertation and a brief summary of included studies and publications (Section 2.7).

2.1 A Definition of Hardware Reverse Engineering

Rekoff (1985) defined reverse engineering as a process through that a reverse engineer retrieves underlying structures, components, and functionalities of an existing system without access to its high-level description. The process of reverse engineering is commonly employed in various domains such as software, electronics (e.g., microchips), mechanical engineering, or in the context of cybersecurity as the two principle types: Hardware and Software Reverse Engineering. HRE is a tool to retrieve information about the inner structure and functionality of microchips and is applied for both legitimate (e.g., detection of hardware Trojans) or illegitimate (e.g., IP infringement) purposes (Fyrbiak et al., 2018b; Quadir et al., 2016) (see Introduction).

In order to analyze an unknown chip design, HRE is typically conducted through the following two stages. First, the analyst performs several steps to retrieve a model of the chip (a so called “gate-level netlist”) that consists of gates and their interconnections (Torrance & James, 2009). In a real-world context, those gate-level netlists consist of thousands or millions of logic components, and can be retrieved from an IC or a Field Programmable Gate Array (FPGA) or from an interception of design information (Becker et al., 2020). Although the retrieval of a gate-level netlist is very effortful and requires a highly specified set of methods, tools and materials, prior research has shown that trained specialists are able to reliably extract a netlist from both ICs and FPGAs (Ding et al., 2013; Moradi, Baranghi, Kasper, & Paar, 2011; Torrance & James, 2009; Ender, Moradi, & Paar, 2020).

Second, after extracting the netlist, the analyst actively combines several different techniques to achieve a given goal, for instance, to find or understand IP blocks, to extract cryptographic keys (Wallat, Fyrbiak, Schlögel, & Paar, 2017) or to detect malicious manipulations. In order to achieve the specific reversing goal, the second HRE phase typically involves human cognitive abilities to recognize specific modules, to identify blocks of interest, to retrieve a detailed understanding of Boolean sub-circuits, or to recover high level register (Azriel et al., 2019; Fyrbiak et al., 2018b; Subramanyan et al., 2014; Albartus, Hoffmann, Temme, Azriel, & Paar, 2020; Becker et al., 2020). Due to the complex design of commercial hardware, analysts usually have to create custom solutions or need to perform detailed manual analysis of single netlist components (Becker et al., 2020; Fyrbiak et al., 2017).

Nevertheless, the tool support in the domain of HRE is not as mature as, for example, in the domain of Software Reverse Engineering (SRE). SRE is regularly applied to use cases such as malware detection, security analysis, obsolescence management, or the realization of interoperability (Eilam, 2011). Therefore, SRE is commonly applied to analyze highly optimized bytecode (i.e., strings consisting of zeroes and ones) that is received from machine code or binary files that run on specific devices (Gomulkiewicz & Williamson, 1996). Software reverse engineers commonly use specific tools to conduct SRE, such as decompilers, disassemblers, low-level debuggers and packet sniffers. In the past decades, several advanced SRE tool suites (e.g., IDA Pro; Ghidra) have been developed and provide software reverse engineers a variety of (semi-) automated functionalities that suite to the analysts' individual reversing goals. The SRE community benefits from long-standing tool development, an active research community, and well-established business cases that have led to the advancement from mostly manual SRE analysis to predominately automated process (Gandotra, Bansal, & Sofat, 2014). In terms of HRE, some developments of specific tool suites have been advanced in the past, such as the HRE tool HAL by Fyrbiak et al. (2018a). HAL, as a uniform platform, offers several functionalities that are essential for conducting HRE (e.g., graph-based visualization of gate-level netlists or an interactive netlist exploration via a Python console).

Although HRE tools like HAL exists, they do not support a fully automated analysis of the gate-level netlist, and hardware reverse engineers are usually forced to make sense out of previously described semi-automated steps. Thus, cognitive processes and abilities of the analysts critically influence and determine the success of HRE. Surprisingly, those underlying cognitive processes that have a huge impact on HRE, have barely been considered by prior research and remain poorly understood. So far, only little research has been investigated to explore and define those underlying cognitive processes in HRE. Within this thesis, I contribute to this research gap by systematically analyzing cognitive processes of hardware reverse engineers. In order to impede HRE, we argue that it may be a valuable approach to develop obfuscation schemes that consider both, the technical and the cognitive processes of HRE – suggested as cognitive obfuscation (Becker et al., 2020; Wiesen et al., 2019a). Based on our findings on HRE problem solving, I develop initial ideas for cognitively challenging obfuscation tasks in the final part of my dissertation.

2.2 Problem Solving in Hardware Reverse Engineering

As shown above, prior research has already investigated general technical analysis methods of HRE. However, HRE also requires custom solutions as well as analysts' skill, knowledge, and cognitive processes. Surprisingly, so far little is known about these cognitive processes that influence HRE performance of human reverse engineers. In the following, I present relevant prior findings on problem solving in reverse engineering of Boolean systems. As more research on cognitive processes in HRE is lacking thus far, I refer to more general psychological research findings on the definition of problem solving and influencing factors such as the level of expertise or the level of cognitive abilities of a problem solver.

Lee and Johnson-Laird (2013) define reverse engineering of Boolean systems, i.e., binary logic expressions, as a specific and strongly understudied type of human problem solving. In their theoretical model, reverse engineering of Boolean systems is specified as:

> „ *[...] the process of working out how to assemble components with known properties into a system that has the input-output relations of a target system*" Lee and Johnson-Laird (2013, p. 20).

In other words, during reverse engineering of Boolean systems, problem solvers infer the underlying mechanisms of a given system, and investigate how single components of the system influence specific outputs, and if these components also depend on each other. In their work, Lee and Johnson-Laird (2013) conducted five laboratory studies with students who had no domain-specific knowledge. During these five experiments, the participating students were asked to draw Boolean circuits that would control electric lights or a water flow system. The authors concluded that reverse engineering of Boolean systems was influenced by three factors: (1.) The number of variable components (i.e., the switches), (2.) their number of settings that yielded the positive outputs (i.e., light comes on), (3.) the interdependence of components that influenced the outputs. Additionally, Lee and Johnson-Laird (2013) collected interview data during their experiments that enabled them to draw conclusions about applied problem-solving strategies of the participating students. The results revealed that the participants chose one of two main initial strategies: They focused either on a single output at a time or on a single component at a time. Moreover, the qualitative data showed that participants extended the solutions they gained from previous sub-problems to the next sub-problem.

It nevertheless remains an open research question whether Lee and Johnson-Laird's (2013) results are applicable to describe problem-solving processes in HRE for two main reasons. First, Lee and Johnson-Laird (2013) included task materials of simple Boolean systems in their experiments that are strongly limited in their comparability to complex real-world HRE tasks. During realistic HRE challenges, hardware reverse engineers have to make sense of hundreds of thousands or millions of logic components included in a gate-level netlist. Second, they recruited students who lacked relevant background knowledge for example in digital circuits, Boolean algebra, or electrical engineering. However, it can be assumed that hardware reverse engineers possess and apply domain-specific knowledge and skills to analyze a real-world gate-level netlist that in turn strongly questions the generalizability of the results of Lee and Johnson-Laird (2013) to the HRE domain. Thus, it is necessary to conduct studies with realistic HRE settings (e.g., realistic task materials; realistic tool support; participants with domain-specific expertise) that closely approximates real-world conditions to systematically explore problem-solving processes in HRE and influencing factors of the human analysts.

In 2020, we (Becker et al., 2020) exploratorily investigated the technical processes of HRE and proposed an HRE model consisting of the following three phases: (1.) Candidate Identification, (2.) Candidate Verification, and (3.) Realization. We suggested that hardware reverse engineers analyzed an unknown gate-level netlist by passing through these three phases. We found that analysts applied a set of both manual analyses (e.g., visual identification of important netlist components) and semi-automated steps (e.g., verification of components based on developed programs or algorithms). Within an exploratory analysis we compared a fast and a slow hardware reverse engineer. Our results showed that the faster participant applied a strategy that could be described as divide-and-conquer, and that the slower participant chose the HRE task through a trial-and-error strategy. Furthermore, we found that the cognitive factor working memory played a role in solving the HRE task time-efficient. I will refer to that result in greater detail in my general discussion (see Chapter 5). In our paper (Becker et al., 2020), we contributed to the understanding of technical processes underlying HRE, but also derived the need for further research. Hence, it is essential to systematically analyze problem-solving processes in greater detail and compare applied strategies of more than two hardware reverse engineers.

As only little prior research on problem solving in HRE exists, I will refer to more general findings of problem-solving research in other domains. Prior psychological research defines problem solving as an essential cognitive ability and a key competence that enables people to solve complex situations in their daily lives (Funke, Fischer, & Holt, 2018). In the beginning of a problem-solving situation, the problem solvers are in the initial state in that they lack the knowledge for producing an immediate and routine solution. In order to bridge the gap between the initial state and the desired goal state, i.e., the problem solution, the problem solvers can apply a set of cognitive operators (e.g., problem-solving strategies) (e.g., Fischer, Greiff, & Funke, 2011; Dörner & Funke, 2017). As suggested by Newell and Simon (1972) it is assumable that problem solving in many domains (and thus also in HRE), may pass through seven stages: Problem categorization; Construction of a mental problem representation; Search for suitable operators (e.g., strategies or procedures); Application of selected operators; Evaluation of solution; Iterative refinement of strategies if solution is not satisfactory; Storage of solution.

In order to describe and define types of problem solving, prior psychological research has established a broad range of taxonomies. The taxonomy of Reitman (1965) that distinguishes between well-defined and ill-defined problems, is widely accepted and applied is. Well-defined problems are characterized by a representation in a problem space, a clearly defined goal state, and a clear set of means enabling the problem solver to reach the goal state (Dörner & Funke, 2017). The Tower-of-Hanoi problem is an example for a well-defined in that the problem solver receives three rods and several different-sized discs. The clearly defined goal is to move the discs from the left to the right rod in ascending order of disc diameter. In order to solve the problem, the problem solver has a clearly defined set of means (e.g., only one disc can be moved at a time; no larger discs can be put on smaller ones).

In contrast to well-defined problems, ill-defined or complex problems cannot be represented in a problem space, and have neither a clear problem definition, nor a clearly defined goal state, nor precisely defined means for solving the problem (Dörner & Funke, 2017). Funke (2012) defines five characteristics of complex problems: 1) complexity, 2) connectivity, 3) dynamics, 4) intransparency, and 5) polytely. More precisely, complex problems consist of multiple and interacting variables (complexity) that are interconnected (connectivity). Moreover, these multiple and interacting variables of the complex problem system change due to the influence of the problem solver or independently (dynamics). Furthermore, the structure of the complex problem (e.g., information about variables, or problem states) is partially or completely intransparent to the problem solver (intransparency) and whereby the problem solver must try to actively gain knowledge about the problem. In order to solve the complex problem, the problem solver has to handle many, and sometime even mutually exclusive goals (polytely). Dörner and Funke (2017) continue in defining complex problems by outlining that complex systems include at least the following three characteristics: 1) there are different levels of abstraction through which a complex system can be described; 2) a complex system includes dynamics that lead to the system's development over time, the system's history and current state, as well as the system's (unpredictable) future; 3) a complex system is knowledge-rich, and the problem solver activates large semantic networks and a wealth of potential problem-solving strategies (domain-specific and domain-general).

The so-called Lohhausen problem is a classic problem system for assessing complex problem-solving performance. In the Lohhausen problem, the problem solver takes over the role of a mayor of a small town and has to manage more than 2000 variables of a dynamically changing system (Dörner, 1980). Prior research defines that barriers between the initial state and the goal state of complex problems state can only be overcome by problem solver who apply non-routine solution methods (Funke, 2012; Mayer, 1992; Mayer & Wittrock, 2006). Furthermore, the problem solver needs to acquire knowledge about the system's variables and structure (i.e., knowledge-acquisition phase; Novick & Bassok, 2005). To transfer the initial state into the goal state, the problem solver applies the previously acquired knowledge (knowledge-application phase, Novick & Bassok, 2005). Based on knowledge-acquisition and application, the problem solver builds a problem representation that leads to the selection of cognitive operators (e.g., problem-solving strategies) (e.g., Mayer & Wittrock, 2006; Novick & Bassok, 2005).

Lee and Johnson-Laird (2013) presented in their theoretical model on reverse engineering of Boolean systems that reverse engineering may be a specific and so far poorly understood type of human problem solving. Unfortunately, the authors did not conduct a typological embedding of reverse engineering in existing research on human problem solving. Thus, it is an open question to which problem-solving taxonomy HRE can be assigned or to what extent HRE problem solving combines aspects of well-defined or ill-defined problems. In order to understand HRE problem solving and also to explore correlations with prior domain-specific knowledge or intelligence, it would be valuable if researchers would derive a theoretical analysis and definition of HRE as a specific type of human problem solving. This embedment would support the psychological community to discuss and explore this specific type of problem solving in greater detail.

2.3 Expertise and Problem Solving

An individual's level of domain-specific expertise has been shown to mainly influence the problem-solving performance in many domains such as chess (Chase & Simon, 1973a, 1973b) or medicine (Lesgold et al., 1988). Unfortunately, practically no research has been conducted on how HRE experts analyze an unknown netlist to retrieve components or if HRE experts and non-experts differ in their problem-solving performance. Although, there are some best practice examples by HRE experts who analyzed unknown netlists (Tarnovsky, 2019; Thomas, 2015), controlled and systematic psychological studies on how HRE experts analyze an unknown chip design are missing thus far. Against this background, I refer to general findings in expertise research to define my research goal on analyzing potential expertise-related differences in HRE.

A wealth of prior research has shown that expertise develops due to years of daily deliberate practice (Ericsson, Krampe, & Tesch-Römer, 1993), and that superior performance by experts is based on domain-specific knowledge (Nokes, Schunn, & Chi, 2010). Other researchers postulate that expertise is related to talent (Galton, 1870), and can be described as a combination of genetic dispositions and experience (e.g., Simonton, 1999). Furthermore, research has shown that differences in superior performance by experts may be influenced by general cognitive ability factors (e.g., intelligence score; working memory capacity), without denying the major influence of domain-specific knowledge and skills (e.g., Hambrick, Burgoyne, & Oswald, 2019; Grabner, Neubauer, & Stern., 2006).

Differences between experts and non-experts in problem solving are based on the experts' domain-specific knowledge that supports experts in solving domain-specific problems faster and more accurately than non-experts (Nokes at al., 2010). Prior research suggests that experts activate domain-specific knowledge and strategies to solve the current problem with low cognitive effort (Alexander, 2003; Chi, 2006) and apply their skills with greater automaticity (Schneider, 1985; Chi, 2006). Ericsson and Kintsch (1995) suggest that experts have developed effective long-term working memories that support experts in retrieval of prior knowledge to solve problems, and also enables experts to circumvent the common limits of the memory. Due to their rich, and well-organized knowledge structures, experts acquire new knowledge from problem solving more easily than novices, and are more efficient in merging novel domain-specific knowledge into already stored knowledge structures (Nokes et al., 2010).

Prior research has revealed several results when experts achieve the best solution in solving domain-specific problems. In the following, I present some examples that outline differences between experts and novices in solving problems. For example, experts percieve and categorize a problem in a way that is different from non-experts. Lesgold and colleagues (1988) showed that medical experts were able to figure out the correct shape and size of abnormalities in medical picture. Compared to those experts, medical novices only identified fractions of the presented abnormalities (Lesgold et al., 1988). Furthermore, a study in the domain of physics revealed that physic experts and novices categorized the same problems on a different level of abstraction, whereas experts categorized the problems on deeper levels of abstraction (Chi, Feltovich, and Glaser, 1981). Similar results on differences in problem perception and categorizations of experts were shown in various other domains such as mathematics (Silver, 1979), computer programming (Adelson, 1981), and engineering design (Moss, Kotovsky, & Cagan, 2006). Nokes and colleagues (2010) summarize that these differences in problem perception and categorization are based on experts' domain-specific well-structured knowledge.

Prior research showed that experts and non-experts also differ in the final stages of problem solving (i.e., solution evaluation and storage). Experts seemed to evaluate longer than novices, whether the solution met the problem requirements (Groen & Patel, 1988; Voss & Post, 1988). Furthermore, during the evaluation phase, experts identified and corrected more errors than novices, and were also more willing to modify their strategies, than novices (Nokes et al., 2010). In contrast, novices proceed with incorrect assumptions and errors, as shown in the context of a historical task (Wineburg, 1998; Nokes et al., 2010).

In summary, a wealth of prior research has shown how expertise influences single stages of a problem-solving process, suggesting that experts solve problems more efficiently than non-experts. It is an open question as to whether problem solving of HRE experts is more efficient than problem solving employed by HRE non-experts. It is assumable, based on prior findings in expertise research, that HRE experts may solve HRE problems more efficient due to their profound level of domain-specific knowledge. Against the background of this chapter, it is highly likely that expert hardware reverse engineers have developed an extensive base of knowledge and set of skills through years of deliberate practice (Ericsson et al., 1993). These profound knowledge structures may enable HRE experts to perceive, categorize, and select suitable strategies more efficiently than HRE non-experts. Such HRE experts may have gone through years of deliberate practice that enabled them to develop rich and well-organized knowledge structures. Consequently, those HRE experts may have been able to produce and store efficient solutions by leveraging their fast problem-solving and procedural work flows (e.g., schemas and chunks stored in long term working memory (Ericsson and Kintsch, 1995)). To frame the point in more everyday terms, an expert reverse engineer may know at once how to overcome an HRE task similar to a situation in which a chess master immediately recognizes which moves are feasible (e.g., Chase & Simon, 1973a) or in which a skilled physician is able to immediately render a diagnosis based on observable symptoms (Lesgold et al., 1988).

2.4 Intelligence and Problem Solving

Besides expertise and prior domain-specific knowledge, prior research has also analyzed the influence of general intelligence or sub-factors of intelligence on problem-solving performance. As I am not aware of any prior research that has investigated the impact of general intelligence or sub-factors of intelligence on the problem-solving performance in HRE, I refer to more general findings on the relation between intelligence and problem solving in the following chapter. As it is also unclear, which aspects of simple and / or complex problems are included in HRE (see Background), I present the relation between intelligence and both types of problem solving.

Prior researchers have established various definitions of intelligence. I quote the working definition by Gottfredson (1997, p. 13) who defined intelligence as followed:

> „*Intelligence is a very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly, and learn from experience. It is not merely book-learning, a narrow academic skill, or test-taking smarts. Rather, it reflects a broader and deeper capability for comprehending our surroundings, ‚catching on', ‚making sense' of things, or ‚figuring out' what to do.“* Gottfredson (1997, p. 13).

Besides several definitions, prior research has also presented a wealth of theories of intelligence. Spearman (1927) defined intelligence as a single trait and proposed the concept of general intelligence (*g*). Other representatives are for example Cattell (1987) with the distinction between fluid and crystallized intelligence, or Thurstone (1938) who proposed a division of intelligence into seven primary factors.

As outlined by Wenke, Frensch and Funke (2005), a wealth of prior research has investigated the relation between simple problem solving and individual intelligence, postulating that if a relation exists it might be on modest size (i.e., r = .30) (e.g., Sternberg, 1982). Subsequently, analyzing problem-solving processes in simple problems was hypothesized as a means of specifying cognitive processes that form intelligence (Resnick and Glaser, 1975).

In terms of complex problem solving (CPS), the relation to individual intelligence is not that clear as it is to simple problem solving, what is also discussed in a meta-review by Stadler et al. (2015). Theoretically, CPS includes several aspects (e.g., the integration of information) that are also part of intelligence definitions (Sternberg & Berg, 1986). But other CPS characteristics such as dynamics or intransparency are not included in definitions of intelligence. This theoretical ambiguity is also reflected in contradictory study results on the relation between CPS performance and intelligence, ranging from non-significant (e.g., Joslyn & Hunt, 1998; Putz-Osterloh, 1985) to strong correlations (e.g., Funke & Frensch, 2007; Wüstenberg, Greiff, & Funke, 2012). For example, the significant differences of participants' CPS performance could not be explained by general intelligence (Brehmer, 1992; Rigas & Brehmer, 1999). Furthermore, Kluwe, Misiak and Haider (1991) summarized in a meta-review that 11 studies failed in showing a close relation between intelligence scores and the performance in CPS tasks. Subsequently, Wenke and colleagues (2005) postulated that convincing empirical data on the relation between global intelligence and CPS performance does not exist (without denying a hypothetical relation between the both constructs). In contrast, other studies found correlations between CPS performance and general intelligence. For example, prior research showed correlations between various measures of CPS and general intelligence between r = .33 and r = .63 (Gonzalez, Thomas and Vanyukow, 2005), or correlations of r = .40 between CPS performance in Tailershop tasks and general intelligence (Süß, Kersting, & Oberauer, 1991; see also Stadler et al., 2005).

In terms of problem solving in HRE, the influence of individual intelligence or sub-factors of intelligence has not been investigated by prior research. Hence, it is unclear if intelligence plays a role in solving HRE tasks, and should be investigated in research.

2.5 Methodological Challenge – Unavailable HRE Experts

Researchers who plan to conduct experimental studies in HRE face the methodological problem that HRE experts are unavailable for research. The very small worldwide population of HRE experts makes the efforts to locate them not only very time-consuming, but also often unsuccessful. The availability of those HRE experts for studies is even more aggravated when one considers that HRE experts often work for highly-specialized companies or government agencies. Those companies and agencies often impose contractual restrictions that forbid HRE experts from revealing details of their working processes. Such otherwise potentially willing and available HRE experts are consequently unable to participate in research that explores HRE processes and underlying cognitive factors.

Prior research in other cybersecurity domains (e.g., developer studies in the context of security software engineering) also faced the methodological problem of recruiting large samples of expert developers that is why usable security researchers often choose to recruit students with relevant backgrounds (besides professionals) (e.g., Acar et al., 2017; Naiakshina, Damilova, Tiefenau, & Smith, 2018). Prior research presents contrary results on the comparability of developer experts and computer science students. Naiakshina et al. (2019) showed that computer science students and freelancers behaved similarly with regard to secure password storage. Acar and colleagues (2017) found that self-reported status as student or professional did not significantly influence the functionality, security, or security perception of written Python code. In contrast, Naiakshina and colleagues (2020) conducted a password-storage study with computer science students, freelancers, and company developers. Their results revealed that company developers performed better with regard to security measures than students and freelancers (Naiakshina et al., 2020). However, Naiakshina and colleagues (2020) do not deny the fact, that including students in cybersecurity studies will reveal valuable insights, but also outline that the effects of different treatments for different participant groups (e.g., prompting for secure password storage) are needed to achieve valuable results with a student sample.

The methodological problem in HRE research raises the question how studies on problem solving in HRE can be conducted when experts are unavailable for research and global populations of analysts are very small. Although the recruitment of students to study HRE problem solving may be a valuable starting point, it nevertheless remains unclear, how to train students in HRE in order to enable them to solve realistic HRE tasks during the study. HRE tasks are highly specific and are commonly not part of any computer science nor cybersecurity study program at universities. There is an almost complete lack of educational courses in the HRE field and HRE training happens almost entirely on the job, which even makes it more difficult to recruit students for our studies. It remains unclear, how a course on teaching HRE has to be designed in order to promote HRE skill acquisition in students that prepares them to solve a realistic HRE task.

2.6 Research Goals

In my dissertation, I aim to accomplish one overarching research goal, namely the analysis of underlying human factors in HRE with a specific view on problem-solving processes and the role of cognitive abilities and expertise. In addition to my main research goal, I establish two sub-goals that emerged from the reviewed literature on human problem solving. Beyond the contributions to the main research goal and the sub-goals, I aim to develop initial ideas for cognitive obfuscation based on the main results. In the following, I will describe the goals of my dissertation in more detail.

Overarching Research Goal: Analyzing Problem-Solving Processes in HRE

It is unclear how hardware reverse engineers proceed in analyzing an unknown netlist and what kind of problem-solving processes and strategies are involved in HRE. Furthermore, it remains unanswered by prior research how efficient the applied problem-solving strategies are and which problems and difficulties hardware reverse engineers face during HRE. In order to accomplish the overarching research goal, we conducted an empirical study with hardware reverse engineers who were asked to solve a realistic HRE task. We analyzed the problem-solving processes in HRE and furthermore, systematically explored applied problem-solving strategies. I discuss the findings from our empirical study on HRE problem solving in the light of established psychological theories on human problem solving and in the light of Lee and Johnson-Laird’s (2013) theory of reverse engineering of Boolean systems.

As shown in the background of the dissertation (see Background), different variables (e.g., expertise) influence problem-solving performance. However, in the area of HRE problem solving, it is unexplored whether cognitive factors of hardware reverse engineers play a role in analyzing an unknown netlist. To address this question, I derive two sub-goals, which are briefly presented below.

Sub-Goal 1: Investigating the Role of Expertise in HRE Problem Solving

Previous research has shown (see Background) that expertise and domain-specific knowledge play a role in several problem-solving domains and often resulted in experts solving a problem more accurately and efficiently than novices or intermediates (e.g., Nokes et al., 2010). Nevertheless, it remains an open question whether this is also true for the HRE domain and if HRE experts are better at analyzing an unknown netlist than non-experts. In order to shed light on this question, the present work recruited hardware reverse engineers on different levels of expertise (i.e., intermediates and expert) to investigate expertise-related differences in HRE problem solving.

Sub-Goal 2: Investigating the Role of Cognitive Abilities in HRE Problem Solving

As shown in the background (see Background), the level of cognitive abilities can influence problem-solving performance (e.g., Gonzalez et al., 2005). However, it is unclear whether cognitive factors such as intelligence or sub-factors of intelligence play a role in HRE problem solving. In order to address this question, we examined whether the level of cognitive abilities was correlated with time-efficient problem-solving performance in HRE.

Development of Initial Ideas for Cognitive Obfuscation

Beyond the contributions to the main goal and the two sub-goals of my doctoral thesis, I aim to develop initial ideas for cognitive obfuscation. As outlined above (see Background), we suggested that sound countermeasures impeding HRE should also consider cognitive processes and cognitive factors of hardware reverse engineers (e.g., Wiesen et al., 2019a). Based on our findings from our study on HRE problem solving, I develop initial ideas for future research in this novel field of cognitive obfuscation methods, which could impede both expertise-related and cognitive abilities-related processes, as both seemed to play a role in HRE problem solving.

2.7 Thesis Overview

The thesis consists of three parts that address the previously established research goals of the dissertation (see Research Goals), with i) preparation part (see Chapter 3), ii) main part (see Chapter 4), and iii) conclusion part (see Chapter 5) including initial ideas for cognitive obfuscation impeding HRE (Figure 1 illustrates the structure of the dissertation.). The thesis combines parts of four papers that comprise our scientific work on problem-solving processes in HRE.

As outlined above (see Background), researchers who plan to conduct studies on cognitive processes in HRE, face the problem that HRE experts are unavailable for research and that global populations of hardware reverse engineers are very small. In order to circumvent this problem and to enable our main study, we suggested, developed and evaluated a methodological approach (preparation part). In order to explore underlying cognitive processes and factors in HRE, we conducted an empirical study and analyzed applied problem-solving strategies, their time-efficiency, and influencing cognitive factors such as the working memory of the participants (main part). Finally, I addressed our main results of the empirical study on human factors in HRE by discussing them in terms of previous findings on human problem solving (conclusion part).

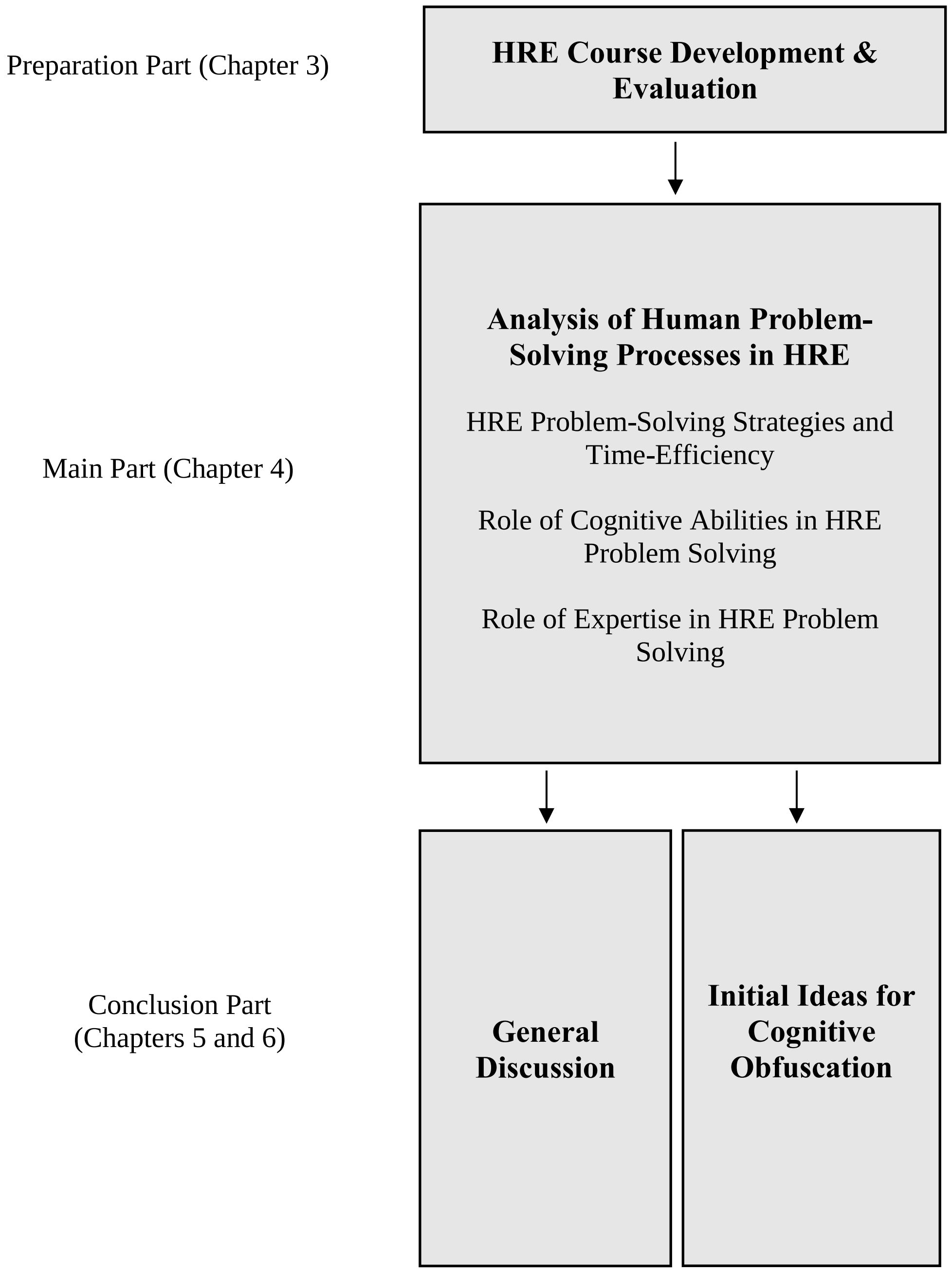

Preparation Part (Chapter 3)
HRE Course Development & Evaluation
Main Part (Chapter 4)
Analysis of Human Problem-Solving Processes in HRE
HRE Problem-Solving Strategies and Time-Efficiency
Role of Cognitive Abilities in HRE Problem Solving
Role of Expertise in HRE Problem Solving
Conclusion Part (Chapters 5 and 6)
General Discussion
Initial Ideas for Cognitive Obfuscation

Figure 1. Overview and structure of this doctoral thesis.

Preparation Part – Circumventing the Methodological Challenge

Sections of the following papers are incorporated into the preparation part of the dissertation (see Chapter 3) that suggests an approach to circumvent the challenge that HRE experts are unavailable for research.

Wiesen, C., Becker, S., Fyrbiak, M., Albartus, N., Elson, M., Rummel, N., & Paar, C. (2018, December). Teaching Hardware Reverse Engineering: Educational Guidelines and Practical Insights. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 438-445). IEEE.

Wiesen, C.; Becker, S., Paar, C., & Rummel, N. (2019). Promoting Skill Acquisition in Hardware Reverse Engineering. In Proceedings of the *2019 IEEE Frontiers in Education Conference (FIE)*, Cincinnati, OH, USA, 2019.

The preparation part comprises our prior work on circumventing the methodological challenge that HRE experts are unavailable for research. In order to solve this challenge, we developed an educational HRE course that enabled students with relevant backgrounds to participate in our main study on HRE problem solving. As HRE was not part of university programs, it was unclear how to develop an HRE course that taught students a sufficient amount of HRE skills, domain-specific knowledge, and specific competencies. As realistic HRE processes commonly involve the analysis of textual or graphical representations of the netlist, we argued that students needed to acquire conceptual and perceptual competencies to work with and learn from these multiple netlist representations. Against a wealth of prior educational and psychological research, we derived specific guidelines for an HRE course design that aimed to promote the acquisition of HRE skills and specific competencies.

Finally, the preparation part of the dissertation presents an evaluation if the course promoted HRE skill acquisition in students with relevant backgrounds. Therefore, two studies at a German and a North American university were conducted that included undergraduate and graduate students enrolled in cyber security and electrical engineering programs. Our results showed that the students were able to achieve high solution probabilities in the first three practical HRE tasks. Nevertheless, the solution probability in the most complex and realistic task 4 decreased significantly (but was still on a satisfactory level). Furthermore, our results showed that students were motivated by the realistic HRE task and the realistic working scenario (e.g., solving the HRE tasks with the tool HAL). We presented our findings in the light of course improvements and suggested that the course would be a sufficient tool to prepare students in participating in our main study.

In summary, the preparation part of the doctoral thesis includes a necessary working step that enabled us to conduct our main study on HRE problem solving by including trained students on intermediate levels of HRE expertise.

Main Part – Analyzing HRE Problem Solving

The main part (see Chapter 4) forms the core of my dissertation as it aims to accomplish the overarching research goal of analyzing the underlying problem-solving processes in HRE. In addition to a detailed and in-depth analysis of HRE problem-solving processes and applied problem-solving strategies, the main part also presents results on the role of relevant cognitive factors such as intelligence and expertise in HRE problem solving.

Becker, S., Wiesen, C., Albartus, N., Rummel, N., & Paar, C. (2020). An Exploratory Study of Hardware Reverse Engineering – Technical and Cognitive Processes. *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, Conference Paper.

Wiesen, C., Becker, S., Walendy, R.; Paar, C., & Rummel, N. (submitted for review; Nov. 2020). *The Anatomy of a Hardware Reverse Engineering Attack: Insights into Cognitive Processes during Problem Solving.* ACM TOCHI

Within the main part of the dissertation, we conducted an experimental study at a German university to analyze problem-solving processes in HRE. As prior research on systematically analyzing applied problem-solving strategies in HRE is limited thus far, we collected behavioral data in form of automatically recorded log files of each participant while they solved a realistic HRE task. In general, prior research suggested that expertise played a role in solving problems (e.g., Simon & Simon, 1978). In order to analyze if expertise and prior domain-specific knowledge were relevant in solving the HRE task, we included participants on different levels of HRE expertise in our study (eight intermediates; one HRE expert). All intermediates acquired necessary domain-specific knowledge and skills through the HRE course evaluated in the preparation part of the dissertation. Following the successful completion of the HRE course, the intermediates were asked to solve the HRE task within 2 weeks using the HRE tool HAL. Additionally, we were able to recruit one HRE expert out of the professional network of one of the co-authors.

Furthermore, prior research postulated that intelligence and specific sub-factors of intelligence influence problem-solving performance (e.g., Stadler et al., 2015). Thus, in order to accomplish second research sub-goal of the dissertation we included a state-of-the art intelligence test (Wechsler Adult Intelligence Scale, Wechsler, 2008) in our study to compare participants with lower and higher intelligence scores in their HRE problem-solving performance.

Using an iterative open coding scheme based on the well-established Grounded Theory method (Strauss & Corbin, 1998), we conducted an in-depth analysis and derived a detailed mapping of the HRE problem-solving process. The results provided a hierarchical taxonomy of HRE problem-solving steps as well as an analysis of applied problem-solving strategies and their time-efficiency. The results showed that all hardware reverse engineers used unique problem-solving strategies and that there seemed to be no single best strategy. The results also revealed that the HRE expert was able to achieve the fastest solution time. However, two intermediates solved the HRE task with solution times that were comparable to the expert's solution time. Furthermore, the intelligence sub-factor working memory seemed to play a role in achieving time-efficient solutions in HRE, as participants with higher levels of working memory tended to solve the HRE task faster than participants with lower scores of working memory.

In summary, the main part of my doctoral thesis accomplishes the overarching research goal by presenting an in-depth-analysis of applied problem-solving strategies and their time-efficiency in HRE. Furthermore, the main part of the thesis demonstrates findings in the light of both research sub-goals 2 and 3 concerning the role of expertise and the role of cognitive abilities (e.g., working memory) in HRE problem solving.

Conclusion Part – General Discussion

Within the conclusion part (see Chapter 5), I discuss our findings on problem solving in HRE and influencing cognitive factors of hardware reverse engineers from two perspectives. First, I discuss our findings in the light of the ongoing debate in psychology questioning the influence of expertise and cognitive abilities on problem-solving performance. Our results revealed that two intermediates with above-average scores in the intelligence sub-factor working memory solved the HRE task with an efficiency that was comparable to the HRE expert. Against this background, I hypothesize that besides expertise, also cognitive abilities may play a role in efficiently solving the HRE task.

Furthermore, the discussion led to several new hypotheses. For instance, I use commonly applied problem-solving taxonomies of simple and complex problems as a theoretical instrument to embed HRE problem solving in established theories and models of human problem solving. I suggest that HRE problems may include aspects of both simple and complex problems, and thus, may be a specific type of human problem solving as previously suggested by Lee and Johnson-Laird (2013).

Beyond the discussion of our findings in the light of my main research goals, I develop initial ideas for cognitive obfuscation that aim to impede the cognitive processes that are involved in HRE (see Chapter 6). Our results suggested, that HRE problem solving may be based on both expertise (i.e., high levels of domain-specific knowledge and profound experience in HRE problem solving), and intelligence (i.e., especially the sub-factor working memory). Consequently, I suggest that during the development of cognitive obfuscation schemes both factors should be considered. For instance, Ericsson and Kintsch (1995) hypothesized that experts were able to circumvent the limited capacity of their working memory by their well-structures long-term working memory. Thus, countermeasures that solely aim to overload the capacity of the working memory may be not sufficient enough for hardware reverse engineers who also possess well-structured knowledge. Against this background, I suggest that cognitive obfuscation should also include obfuscation tasks that aim to create a challenge for analysts with higher levels of HRE expertise and prior experience.

3 Promoting Skill Acquisition in Hardware Reverse Engineering: Educational Guidelines and Practical Insights

Disclaimer: *The content of this chapter concerning educational and psychological guidelines (Introduction, Educational Research as the Foundation for an HRE Lab Course, Guidelines for a course design) was previously published as parts of the paper "Teaching Hardware Reverse Engineering: Educational Guidelines and Practical Insights" that was submitted in the Proceedings of 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering. This paper was written together with my co-authors Steffen Becker, Marc Fyrbiak, Nils Albartus, Malte Elson, Nikol Rummel, and Christof Paar. I have reformulated the respective parts and partly also rearranged them.*

Furthermore, the content of this chapter concerning the course structure and the course evaluation (methods, results, discussion and implications for future HRE course designs) was previously published as parts of the paper "Promoting Skill Acquisition in Hardware Reverse Engineering" that was submitted in the Proceedings of 2019 IEEE FIE Frontiers of Education Conference. This paper was written together with my co-authors Steffen Becker, Nils Albartus, Christof Paar, and Nikol Rummel. I have reformulated the respective parts and partly also rearranged them. Please note, that some sub-parts of the papers were not relevant for the overall argumentation of the dissertation, and were not included. Furthermore, I included an additional background section within this chapter. The additional educational and psychological background is essential to derive guidelines for the development of an HRE course that aims to promote HRE skill acquisition in students.

As all parts of the paper were conducted with my co-authors, I will use the academic "we" to highlight this fact.

3.1 Introduction and Contributions

As outlined in the background of this thesis (see Background), researchers who aim to conduct studies with hardware reverse engineers, face the problem that HRE experts are unavailable for research. One approach to extend the number of reverse engineers, who may be potentially willing to participate in our study on HRE problem solving, is to promote HRE skill acquisition in students with relevant backgrounds. The inclusion of students in security studies when professionals are unavailable was considered as a valuable approach in related security domains (e.g., Naiakshina et al., 2018). As our studies on HRE problem solving included a realistic HRE task, it was essential to support students in acquiring skills and knowledge in HRE, which prepared them to solve realistic HRE tasks. Therefore, we suggested to develop an HRE course that was influenced by realistic conditions of HRE practice (e.g., reversing tasks; tool support).

As there existed an almost complete lack of educational courses in HRE and structural guidance on how to teach HRE skills, we reviewed findings from educational research (e.g., learning with multiple graphical representations) and cognitive psychology (e.g., skill acquisition based on declarative and procedural knowledge). Based on this foundation, we proposed structural guidelines for an HRE course design that would effectively support students with relevant backgrounds in acquiring HRE skills. Finally, we conducted an evaluation of the effectiveness of the course in teaching HRE skills.

In summary the contributions of this chapter are:

- **Educational Guidelines:** To the best of our knowledge, we were one of the first who established structured guidelines derived from prior psychological and educational research on skill acquisition and learning with graphical and textual representations to develop an HRE course. These guidelines formed the basis for our educational course that aimed to promote HRE skill and knowledge acquisition with a particular focus on gate-level netlist reverse engineering. As the analysis of both graphical and textual representations of a netlist was essential in HRE, our guidelines incorporated structures to support learning from and working with both types of netlist representations.
- **HRE Course Structure:** Based on the proposed guidelines, we developed an HRE course. This HRE course consisted of two phases focusing the acquisition of declarative knowledge (lecture phase) and of procedural knowledge (practical phase with four different HRE tasks). Specific instructional and learning principles were included to promote students' acquisition of specific competencies (e.g., perceptual competencies)

that would support them in learning from and working with multiple netlist representations. Furthermore, students were asked to solve the HRE tasks of the course by using the HRE tool HAL (Fyrbiak et al., 2018a).

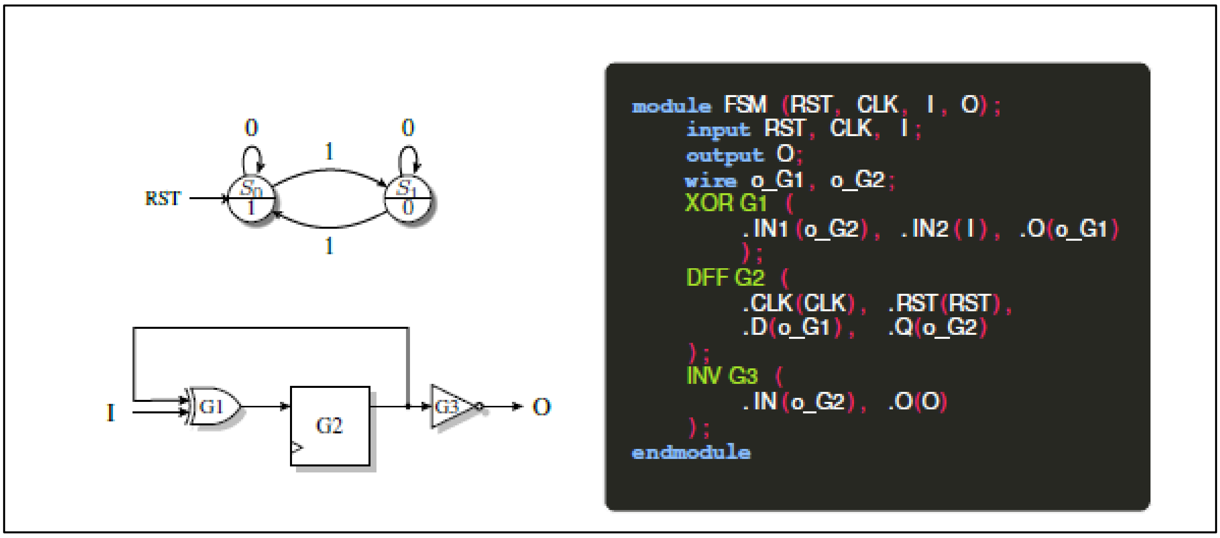
- **Evaluation of HRE course.** We systematically evaluated if our HRE course enabled students to acquire HRE skills within two exploratory studies at a German and a North American university in winter term 2018/2019. Finally, we presented our research questions, methods, and findings of our evaluation study.

3.2 Background and Guidelines

First, we will present relevant background from educational and cognitive psychology that may be relevant for teaching students to work with graphical and textual netlist representations during an HRE course. This includes prior research findings in i) learning from and working with multiple representations; ii) promoting skill acquisition; iii) enhancing students' motivation. Besides these theoretical constructs, we will outline specific instructional principles that are hypothesized by prior researchers to promote specific competencies that enable students to learn from specific representations. Based on this theoretical background, we will derive specific guidelines for an HRE course design.

3.2.1 Background on Learning with Multiple Netlist Representations

Under realistic conditions, hardware reverse engineers have to make sense of various representational forms of a gate-level netlist. Figure 2 (from Wiesen et al., 2018) provides textual and graphical representations of a netlist. The human analyst is forced to obtain the correct information from both sources in order to achieve their reversing goal. Hence, a course that aims to promote skill acquisition in HRE has to consider the integration of both representational forms of a gate-level netlist. From an educational point of view, the retrieval of information from both representations, can pose certain challenges to reverse engineers, and especially to novices who have to learn how to work with and learn from HRE-specific representations (e.g., Rau, 2017). Within the following section, we present relevant background in educational and psychological research that provides guidance on how these challenges can be met.



```
module FSM (RST, CLK, I, O);
    input RST, CLK, I;
    output O;
    wire o_G1, o_G2;
    XOR G1 (
        .IN1(o_G2), .IN2(I), .O(o_G1)
        );
    DFF G2 (
        .CLK(CLK), .RST(RST),
        .D(o_G1),  .Q(o_G2)
    );
    INV G3 (
        .IN(o_G2), .O(O)
    );
endmodule
```

Figure 2. Example Moore Finite State Machine (FSM) circuit as a state transition graph (upper left) with associated gate-level netlist in (1) visual graph-based representation (lower left), and (2) textual representation with an exemplary gate library in Verilog.

Opportunities and Challenges – Learning with Graphical Representations

Graphical representations can support students' learning success by making abstract concepts more accessible (Schnotz, 2014; Rau, 2017), and by depicting supplementary information that enables students to build a deeper understanding of novel content (Ainsworth, 2006, 2014; Rau, 2017). Despite their value in supporting students' learning processes, graphical representations can also be challenging for students. Prior research showed that especially the so-called representation dilemma posed such a challenge (Ainsworth, 2006; McElhaney, Chang, Chui, & Linn, 2015; Rau, 2017). According to Rau (2017), students face a representation dilemma when they are asked to learn new content knowledge that they do not yet understand from graphical representations that they also do not yet understand. In order to overcome representation dilemma and to benefit from the inclusion of graphical representations as parts of the educational program, it is essential that students develop specific competencies (e.g., Rau, 2017).

In the following section, we present an overview of these competencies. We describe in which ways they may support students in recognizing how graphical representations depict relevant information to solve a task or to acquire new skills and knowledge. Furthermore, we present prior research findings the development of these specific competencies may be supported.

Competencies for Learning from and Working with Graphical Representations

In order to overcome the representation dilemma, prior research has suggested that students need to acquire representational competencies that are defined as skills and knowledge that support students to use and learn from graphical representations to solve the task (Gilbert, 2005; 2008; Rau, 2017). In this context, prior research distinguishes between two types of representational competencies: (1) conceptual competencies, and (2) perceptual competencies (Rau, 2017). It is suggested that these two types of representational competencies are acquired through different types of learning processes and thus, through different types of instructional support (Goldstone, 1997; Kellman & Massey, 2013; Koedinger, Corbett, & Perfetti, 2012; Rau 2017).

Conceptual Competencies. Conceptual competencies are defined as knowledge and skills that enable students to relate graphical representations to prior knowledge or to identify and choose the most suitable parts of a graphical representation with the riches amount if relevant information that supports them in solving the present task (Rau, 2017). Koedinger and colleagues (2012) have suggested that students develop conceptual competencies through sense-making processes. In general, sense-making is defined as a process in which an individual combines various information and ideas in a meaningful way to solve a present task (Dougherty and Drumheller, 2006). Prior research suggests that sense-making processes are verbally mediated as students are asked to verbally explain how a graphical representation includes information (Rau, 2017; Koedinger et al., 2012; Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Gentner, 1983).

Perceptual Competencies. Besides conceptual competencies, perceptual competencies are also suggested to be relevant to overcome the representation dilemma (Rau, 2017). Perceptual competencies are defined as the ability that enables a student to immediately and effortless detect the meaning of a graphical representation and to identify visual patterns (Gibson, 1969; Rau, 2017). Perceptual competencies include the concept of fluency in recognizing and processing visual information from and about the graphical representation to solve the task at hand (Airey & Linder, 2009; Kozma & Russell, 2005; Rocke, 2010; Wertsch & Kazak, 2011; Rau, 2017). Koedinger and colleagues (2012) argue that perceptual competencies are acquired through inductive learning processes that are defined as nonverbal automatic processes and are relevant for visual pattern recognition (Rau, 2017). Furthermore, Koedinger and colleagues (2012) suggest that students acquire perceptual competencies while they learn to discriminate, classify, and categorize information (Rau, 2017). As perceptual competencies are nonverbal processes, prior research argues that perceptual competencies may be acquired by experienced-based learning with numerous examples, and not by direct instructions (Gibson, 1969; 2000; Kellman & Massey, 2013; Richman, Gobet, Staszewski, & Simon, 1996; Rau, 2017).

Competencies for Connection-Making Abilities

Besides representational competencies, students may also benefit from connection-making abilities that enable them, for example, to connect multiple several visual representations, to identify and explain similarities or differences between visual representations, or to connect between textual and visual representations (Rau, 2017). As outlined above, both textual and graphical representations are essential in gate-level netlist reverse engineering as both types of representations depict relevant information (see Figure 2). The learning processes that support the acquisition of connection-making abilities are described as verbally mediated sense-making processes that use graphical representations in authentic tasks (Rau, 2017).

In summary, HRE processes are based on working with textual and graphical representations of gate-level netlists. Consequently, a course promoting skill acquisition in this field should integrate these two forms of representations. Prior research has shown that learning with both representations can be beneficial and challenging at the same time, described as the representation dilemma (e.g., Rau, 2017). In order to overcome the representation dilemma, students need to acquire conceptual and perceptual competencies that can be acquired through different learning processes and instructional support (e.g., Koedinger et al., 2012). Based on this, we propose four guidelines for an educational course that teaches HRE.

3.2.2 Guidelines for Learning HRE with Multiple Netlist Representations

Guideline 1 – Integration of Graphical Representations

As outlined before, working with graphical representations of a gate-level netlist is a common practice in HRE. Furthermore, prior research has shown that the integration of graphical representations in an educational program can be beneficial for students – especially for those students with lower levels of prior domain-specific knowledge (Kalyuga & Singh, 2015; Mayer & Feldon, 2014; Rau, 2017), and if the graphical representation includes further relevant information (Schnotz & Bannert, 2003; Rau, 2017). Thus, domain-specific graphical representations of gate-level netlists as parts of tasks and exercises, should be an integral part of a course that teaches HRE and prepares students in the best possible way to solve realistic HRE tasks.

Guideline 2 – Support the Development of Conceptual Competencies

Beside the presented benefits, learning from and working with graphical representations can pose a challenge for students, especially for those who are using representations for the first time (Rau, 2017). In order to overcome the representation dilemma, prior research suggests specific instructional support to promote the acquisition of conceptual competencies (Rau, 2017). According to Koedinger and colleagues (2012) the acquisition of conceptual competencies is based on verbally-mediates sense-making processes. These sense-making processes are hypothesized to be activated by instructional support that engages students in active reasoning, for example by prompting students to self-explanations about how they applied a specific graphical representation or by engaging them in discussion with other students about how to solve the task at hand (Koedinger et al., 2012). Hence, a course on teaching HRE should support students' development of conceptual competencies by including specific instructional principles such as self-explanations.

Guideline 3 – Support the Development of Perceptual Competencies

Besides conceptual competencies, perceptual competencies are suggested to support students in overcoming the representation dilemma (Rau, 2017). In order to prepare students to immediately recognize visual patterns and to effectively process information depicted from a graphical representation, it is important to support the development of students' visual fluency (Airey & Linder, 2009; Kozma & Russell, 2005; Rocke, 2010; Wertsch & Kazak, 2011; Rau, 2017). As suggested by prior research, inductive learning processes may support the acquisition of perceptual competencies (Koedinger et al., 2012). Exercises, such as experienced-based learning with numerous examples is suggested to support the acquisition of visual fluency and perceptual competencies (Gibson, 1969; Kellman & Massey, 2013; Richman, Gobet, Staszewski, & Simon, 1996; Rau, 2017). Thus, an educational course on teaching HRE should involve such repetitive exercises of working with graphical representations of gate-level netlists to support students in developing visual fluency for detecting relevant patterns therein.

Guideline 4 – Support to Develop Connection-Making Abilities

As shown in Figure 2 above, gate-level netlists can be represented both graphically and textually. Against this background, it is essential that an HRE course supports students in the acquisition of connection-making abilities within different netlist representations and between the textual and graphical representations of netlists (Rau, 2017). According to Rau (2017) connection-making abilities are mediated by verbal sense-making processes. Thus, an HRE course should prompt students by specific instructions to actively reflect and explain their solutions verbally during lectures.

3.2.3 Background on Skill Acquisition

With the goal of designing the HRE course to enhance the acquisition of HRE skills, we acknowledged the distinction between declarative and procedural knowledge acquisition by assuming that knowledge is first acquired declaratively, and is then transformed into a procedural form (Anderson, 1982).

Prior research has brought many findings to light how both knowledge types are acquired. According to the Adaptive Control of Thought-Rational (ACT-R) (Anderson, 1982), knowledge is represented in two ways (Tenison & Anderson, 2016): Declarative knowledge that consists of facts or strategies (i.e., knowing that), and procedural knowledge that consists of specific actions and procedures of how to achieve goals (i.e., knowing how) (e.g., Nokes et al., 2010). The acquisition of declarative (verbal) and procedural (non-verbal) knowledge is supported by various learning processes (Koedinger et al., 2012). Prior research suggests that declarative knowledge is acquired through understanding and sense-making processes that involve verbally-mediated (oral and written) and explicit processes in which students attempt to understand and reason (Chi & Ohlsson, 2005; Nokes et al., 2010). Students need to be actively engaged in understanding and reasoning (Koedinger et al., 2012). Specific processes are hypothesized to support learning processes of declarative knowledge, for example, comprehension, analogy building, or self-explanation (Chi & Ohlsson, 2005; Nokes et al., 2010). Procedural knowledge (or skill) is suggested to be acquired through repeated practice of a specific task (Anderson, 1982, 1987).

The learning mechanism behind procedural knowledge is knowledge compilation (Anderson, 1987). According to Anderson (1987), declarative knowledge is compiled or combined into a set of procedural rules to solve a specific problem. With practice (repetitions in solving specific problems), these context-specific procedures are concentrated or chunked together (Anderson, 1987). These sets of skills can be quickly and efficiently applied in order to solve a problem (Anderson, 1987). A learning mechanism for knowledge compilation is defined as memory and fluency-building processes (Koedinger et al., 2012). Memory and fluency-building processes are characterized by nonverbal learning processes that involve strengthening memory and compiling knowledge (Koedinger et al., 2012). Fluency is defined as the ability to quickly and accurately solve a problem (Kilpatrick, Swafford, & Findell, 2001), as knowledge is composed and automatically accessible (Singer-Dudek & Greer, 2005; Skinner, Fletcher, & Henington, 1996). Prior findings showed that students with high scores in fluency maintain their skills over time (Singer-Dudek & Greer, 2005) and perform better on more complex tasks than students with lower fluency (Skinner et al., 1996). The learning mechanism of spacing and testing (Pashler et al., 2007; Koedinger et al., 2012) is suggested to support non-verbal memory and fluency-building processes, and thus, the acquisition of procedural knowledge.

We suggest to design the HRE course based on general cognitive psychological research on skill acquisition (Anderson, 1982) that is supported by certain types of instructions and assignments as described. To achieve this goal, we derive two guidelines for designing the HRE course.

3.2.4 Guidelines for the Acquisition of HRE skills

Guideline 5 – Support the Acquisition of Declarative Knowledge (Lecture Phase)

As suggested by prior research, various verbally-mediated (oral and written) and explicit learning processes lead to the acquisition of declarative knowledge (Chi & Ohlsson, 2005; Nokes et al., 2010). Thus, a course that aims to promote HRE skill acquisition should include specific instructional principles that facilitate the learning of declarative HRE facts, theories, and concepts. Therefore, we suggest to include prompted self-explanation (Koedinger et al., 2012) and accountable talks (Michaels, O'Connor, & Resnick, 2008; Koedinger et al., 2012) that are related the acquisition of domain-specific declarative knowledge.

Guideline 6 – Support to Transform Declarative Knowledge into Procedural Knowledge (Practical Phase)

Prior research showed that procedural knowledge is acquired based on knowledge compilation (Anderson, 1987). Knowledge compilation is related to specific learning mechanisms such as non-verbal memory and fluency-building processes (Koedinger et al., 2012). In order to support students in compiling their declarative HRE knowledge and to become more fluently in solving HRE specific tasks, the HRE course should include the learning mechanism of spacing and testing (Pashler et al., 2007; Koedinger et al., 2012). Additionally, non-verbal processes such as induction and refinement processes (e.g. Worked Examples) may improve the accuracy of knowledge through generalization, categorization, discrimination, or causal induction (Koedinger et al., 2012).

3.2.5 Background on Supporting Students' Motivation

Furthermore, motivation is often described as a central driver of devoting years to deliberate practice and learning (Litzinger, Lattuca, Hadgraft, & Newstetter, 2011). A high level of motivation leads to more cognitive engagement, more learning, and higher levels of achievement (Pintrich, 2003), and is therefore relevant to the development of a course on HRE that supports students' learning processes. Since motivation is a key element in learning, we suggest to employ the following design principles in our course to enhance students' motivation by the following guideline for the development of an HRE course.

3.2.6 Guideline for Enhancing Students' Motivation in Learning HRE

Guideline 7 – Support Students' Motivation over the HRE Course

Prior research suggested several mechanisms that may raise and promote students' motivation. As outlined by Litzinger and colleagues (2011) or Pintrich (2003), higher levels of motivation may lead to greater cognitive engagement and learning, when tasks and materials cater both personal and situational interest. Hence, exercises and task of the HRE course should be stimulating and engaging (e.g., by including realistic HRE challenges such as finding control logic in a gate-level netlist). Moreover, motivation may be promoted by the integration of realistic materials and tasks (Ambrose, Bridges, DiPietro, Lovett, & Norman, 2010), such as the HRE tool HAL that combines realistic graphical and textual representations of a netlist. Both, the authentic HRE tasks and the application of a realistic HRE tool may build connections to students' intended profession, and thus, may increase the perceived value of the learning experience, which again may lead to enhanced motivation (Ambrose et al., 2010). Pintrich (2003) postulates that students who believe they are able to solve the present task are stronger motivated in terms of effort and persistence. Thus, it is essential for the development of the HRE course to consider tasks that are on an appropriate level of difficulty.

3.3 Summary of Guidelines and Description of HRE course design

In the following, we provide structural details of the HRE course and indicate how each aspect of the HRE course complies with the previously established guidelines. Table 1 summarizes derived guidelines for the HRE course and how these can be met by for example specific task instructions.

HRE Course Structure and HRE Tasks

Our goal was to develop an HRE course that promotes HRE skill acquisition in students with relevant background (e.g., cyber security; computer science; electrical engineering), and that prepares them in participating in our HRE main study. We divided the HRE course in two phases – a lecture phase and a practical phase. Thus, we followed the ACT-R-model (Anderson, 1982) that showed how skills were acquired. The lecture phase of the HRE course aimed at the acquisition of declarative HRE knowledge; the practical phase at the transformation of declarative into procedural HRE knowledge (skills). During the HRE course students acquired declarative knowledge by verbally-mediated facts, theories, and concepts related to the relevant fields of electrical engineering, Boolean algebra, and graph theory through two 90-minute lectures and one homework assignment per week. We aimed to achieve the acquisition of declarative knowledge through the integration of verbally-mediated exercises that encouraged students to explain their solutions to other students in accountable discussions. Following the lecture phase, students participated in the practical phase consisting of four HRE tasks (detailed descriptions in methods). By including spacing and testing exercises (Pashler et al., 2007; Koedinger et al., 2012), and worked examples (Sweller & Cooper, 1985), we aimed to enhance their long-term retention and improve their fluency in solving HRE tasks. Moreover, through the incorporation of worked examples (Sweller & Cooper, 1985) into the curriculum we aimed to enable students to learn more robustly from tasks that are interleaved with problem solving practice (Koedinger et al., 2012).

Furthermore, the course consisted of five HRE tasks with growing complexity that had to be completed individually by each participant. Each task lasted 2-3 weeks and contained the following subtasks: (1) reading of domain-specific scientific papers, (2) pen & paper exercises, and (3) practical reverse engineering tasks. By reading and understanding of 1-2 scientific papers students learned domain-specific content knowledge for subsequent tasks while the pen & paper exercises supported them in reproducing and internalizing the content knowledge. The acquired content knowledge was first applied to small-scale examples, and subsequently in more complex contexts during the practical reverse engineering tasks (3). At the beginning of each HRE task, theoretical and practical background was taught in one introductory session and after the submission deadline, solutions were discussed and students were encouraged to present their solutions to the class.

Relevance with Respect to Guidelines. The course design consisted of several exercises and HRE tasks that aimed to support students in developing perceptual competencies as defined in guideline 3 (e.g., through repeated tasks involving the use of graphical representations). Furthermore, the course structure satisfied guideline 2 and the development of conceptual competencies, by encouraging active discussions during the lab sessions or presentation by students who explained their solutions. The lecture phase that may promote the acquisition of declarative knowledge through specific learning mechanisms contributed to guideline 5. Both, the integration of spacing and testing, as well as worked examples may also support students in applying their declarative knowledge to solve the four practical HRE tasks, what may lead to the transformation of declarative into procedural knowledge. The practical phase of the HRE course therefore contributed to the fulfillment of guideline 6.

Learning and Working with realistic HRE tool HAL

In cases where higher levels of motivation are associated with greater cognitive engagement and learning (Litzinger et al., 2011; Pintrich, 2003), tasks and materials must cater to both personal and situational interest. The HRE tool HAL (Fyrbiak et al., 2018a) is commonly applied in HRE practice and assists reverse engineers in analyzing complex gate-level netlists.

As suggested by prior research (Ambrose et al., 2010), realistic and authentic materials and tasks could enhance students' level of motivation. Thus, we included the realistic HRE software HAL and realistic HRE tasks in our HRE course. In this context, we aimed to increase the perceived value of the students' learning experience, what again could lead to enhanced motivation (Ambrose et al., 2010).

Relevance with Respect to Guidelines. The inclusion of HAL supported the demands of guideline 1 as HAL provided a learning environment that integrated realistic graphical and textual representations of gate-level netlists. Furthermore, the realistic HRE tool HAL and the realistic HRE tasks contributed to guideline 7 in raising the students' level of motivation that may lead to greater cognitive engagement and learning.

Table 1. Overview about HRE course requirements and guidelines about how requirements can be met.

HRE Course Requirements	Guidelines for HRE course	
Overcome representation dilemma	Promote working with domain-specific representations	*Guideline 1:* Include HRE-specific graphical representations in HRE course
	Conceptual competencies (e.g., Rau, 2017)	*Guideline 2:* Promote acquisition of conceptual competencies by promoting students' sense-making processes (e.g., inclusion of instructional principles such as self-explanations in course) (Koedinger et al., 2012)
	Perceptual competencies (e.g., Rau, 2017)	*Guideline 3*: Promote development of students' perceptual competencies with focus on promoting students' visual fluency in recognizing relevant patterns in HRE netlists by repetitions of numerous HRE examples (e.g., Kellman & Massey, 2013)
	Connection-making abilities (e.g., Rau, 2017)	*Guideline 4*: Support students' connection-making abilities in recognizing how textual and graphical representations complement each other in HRE by activating sense-making processes with instructions prompting students to actively reflect about and explain their solutions (Rau, 2017).

Promote HRE skill acquisition	Acquisition of declarative knowledge (ACT-R; Anderson 1982)	*Guideline 5*: Promote the acquisition of declarative knowledge by promoting verbally-mediated (oral and written) and explicit learning processes such as prompted self-explanation and accountable talks (e.g., Koedinger et al., 2012)
	Transformation of declarative knowledge into procedural knowledge (ACT-R; Anderson 1982).	*Guideline 6*: Promote the acquisition of HRE skills by focusing on students' HRE knowledge compilation (Anderson, 1987) by promoting memory fluency-building processes (e.g., spacing and testing) (Pashler et al., 2007; Koedinger et al., 2012) or by induction and refinement-processes (e.g., worked examples) (Sweller & Cooper, 1985)
Raise students' motivation in acquiring HRE skills	Higher levels of motivation may lead to greater cognitive engagement and learning (e.g., Litzinger et al., 2011)	*Guideline 7*: Enhance students' motivation by including authentic HRE task and realistic HRE working scenarios (e.g., HRE tool HAL) to raise the perceived value of the learning experience (Ambrose, et al., 2010).

3.4 Methods

In order to evaluate if our course enabled HRE skill acquisition, we conducted two studies with students from one German and one North American university. We formulated the following research questions (RQs).

1. Does students' performance in solving HRE tasks improve with increasing experience in HRE?
2. How does the HRE course affect the students' levels of motivation over all HRE tasks?
3. Do the students perceive the growing complexity of the four HRE tasks?

Participants

One study was conducted with 18 participants (mean age *M* = 23.1, *SD* = 1.8; 9 undergraduates) who studied cyber security or electrical engineering at a German university. Overall 20 students (mean age *M* = 23.5, *SD* = 2.3; 9 undergraduates) who were enrolled in electrical engineering or computer science programs participated in the second study that was conducted at a North American university. Five participants had to be excluded because they did not complete all tasks and the amount of data was not sufficient for analysis. Both studies were completed in the winter term of 2018/19, and all participants provided written informed consent beforehand. All participants were recruited from the HRE course in that the studies were embedded. Since, all participants from both universities had a similar study procedure (i.e., similar tasks; similar materials) we were able to merge the two samples in our analyses. The institutional review board (IRB) of the North American university approved the study. Both universities were selected due to their strong programs in cyber security, and computer engineering. In both studies, the participating students received a monetary compensation for the invested time in answering study related surveys and tests. The participants' privacy was ensured by randomly assigned pseudonyms in both studies. Participants consistently used these pseudonyms instead of their clear names throughout all materials and procedures regarding the two studies.

Materials

HRE tool HAL

The HRE tool HAL (Fyrbiak et al., 2018a; Wallat et al., 2019) served as the underlying educational environment for the HRE tasks of the practical phase of the HRE course. Commonly, HAL assists hardware reverse engineers in analyzing complex gate-level netlists and allows for the development of custom plugin by its extensibility (Fyrbiak et al., 2018a; Wallat et al., 2019). In particular, HAL provides both textual and graphical representations of the gate-level netlist under inspection that are depicted in an interactive Graphical User Interface (GUI). The graphical representation of a netlist enables analysts to perform detailed manual inspections and highlighting of crucial netlist components. Furthermore, HAL provides an integrated Python shell to interact with and process the netlist via aforementioned plugins.

HRE Tasks

The practical phase of the course consisted of four HRE tasks, of which each contained the following sub-tasks: (1) the reading of relevant scientific papers, (2) pen & paper exercises, and (3) practical reverse engineering tasks. In order to create HRE tasks that included realistic reversing goals and sub-tasks, the individual HRE tasks were discussed and optimized in advance with representatives from industry.

In the following, we describe the single HRE tasks with special emphasis on the practical tasks. All practical HRE tasks included flat FPGA netlists and had to be solved with HAL. Those netlists did not have any high-level information such as variable and signal names, comments, hierarchies, or module boundaries. Furthermore, the netlists consisted of global input and output buffers, Look-Up Tables (LUTs) and Multiplexers for combinational logic, Flip-Flops for sequential logic and their interconnections. Please note, that hereafter all of those netlist components are simply referred to as gates.

The HRE tasks were characterized by an increasing difficulty due to an increasing complexity of the tasks themselves (e.g., growing number of netlist components), and a decreasing level of guidance by the lecturers and instructional support in the single task sheets (see Table 2; adapted from Becker et al., 2020).

Table 2. Levels of difficulty, netlist complexity, number of netlist components and guidance for each HRE task ranging from + (low) to +++ (high). (*Adapted from Becker et al., 2020)*

	Difficulty Level of Tasks	Netlist Complexity	Number of netlist components	Level of Guidance
HRE Task 1	+	+	131	+++
HRE Task 2	++	+	138	+++
HRE Task 3	++	+	128	++
HRE Task 4	+++	++	2176	++

HRE Task 1 – Introduction to Gate-level Netlist Reverse Engineering: The goal of this task was to introduce the HAL environment and its basic features to the students. In the practical part of this task, students were asked to analyze the data path of an unknown substitution-permutation-network called ToyCipher. Therefore, students identified the block and key sized of the cipher that served as a basis to decide if the implementation was round-based or unrolled. Furthermore, they had to identify S-Boxes. HRE task 1 was characterized by a relatively low complexity with 131 netlist components and straightforwardness of the task itself, and could be solved mostly by manual inspections of the graphical netlist representation in HAL.

HRE Task 2 – Control Logic Reverse Engineering: The goal of this task was to teach students the understanding and implementation of methods used for semi-automated extraction of a control logic – Finite State Machine (FSM). This task included a slightly modified variant of the ToyCipher from HRE task 1. Students were asked to reverse engineer the control logic of the netlist by identifying the logic gates of an FSM. Therefore, students applied graph-based analysis and manual inspection of relevant netlist components. The basic functionality as well as the complexity with 138 netlist components was comparable to HRE task 1.

HRE Task 3 – Reverse Engineering of Obfuscated Control Logic: The goal of HRE task 3 was to teach students how a netlist is protected (i.e. hardware obfuscation) and how to break such a protection. The netlist in this task consisted of 128 gates and was a second variant of the ToyCipher. Furthermore, the underlying netlist included the control flow obfuscation method Harpoon (Chakraborty & Bhunia, 2009). In this context, obfuscation is defined as a transformation that obstructs high-level information without changing functionality (Wiesen et al., 2019a) with the goal of impeding HRE. The HRE task 3 asked students to extract the gates that implemented the control logic and to analyze the obfuscation method. Therefore, students had to differentiate which gates were part of the obfuscation and which gates belonged to the original netlist. Finally, students disabled the obfuscation by patching the initial state and by verifying their results through dynamic analysis of the netlist. The basic functionality and the complexity of the underlying netlist were comparable to the previous tasks. Additionally, HRE task 3 focused on the acquisition of an understanding of obfuscated control logic and on conducting dynamic netlist analysis.

HRE Task 4 – Advanced Encryption Standard (AES) Key Extraction: The goal of the final HRE task 4 was to extract a hardcoded key from a netlist implementing a real-world AES design, whereas AES is a widely applied encryption algorithm. The first sub-task was to retrieve high-level information (about e.g., functionality, the presence of the key schedule, the key length, and the hardware architecture) from this substantially more complex netlist with 2176 gates. As S-Boxes serve as a potential anchor for attacks on hard-coded keys, the second sub-task asked students to write a script to identify the S-Box logic. Based on this, students extracted the hard-coded key through manipulation and dynamic analysis of the underlying netlist. In order to solve the fourth HRE task, students had to apply HRE skills and knowledge they had acquired from previous HRE tasks. Such knowledge and skills were for example, the derivation of high-level information, identification of functional blocks, and dynamic netlist analysis.

Measures & Instruments

Solution Probability

In the studies, we focused on observing changes to and influences on the dependent variable solution probability accuracy in the task (solution probability). The calculation of the solution probability was based on the participants' grading they received for the four HRE tasks of the HRE course. The per-task solution probabilities were standardized as percentage to enable comparison. The resulting scores were the basis from that we calculated the per-task solution probabilities. The scores from every task were standardized as percentage to enable comparison on a scale of 0% to 100%. Three teaching assistants and the lecturer collaboratively graded the participants' solutions from both studies using a detailed gradebook with sample solutions.

Solution Time

The computation of the solution time was based on the log files, which were automatically recorded by HAL. These log files included behavioral data of each student and a time stamp for each activity in HAL. The students' solution times were calculated per HRE task. In order to calculate the time spent on task accurately, we defined an inactivity threshold of one hour and subtracted time periods that lasted longer than one hour from the total solution time.

Further variables of interest

Control Variables

A self-developed questionnaire on socio-demographics asked participants to provide information about their age, major, and target degree.

Questionnaire on Current Motivation (QCM)

In order to investigate the students' level of motivation in solving the single HRE tasks, we employed the Questionnaire on Current Motivation (QCM) (Rheinberg, Vollmeyer, & Burns, 2001). The 18 items of the QCM measured four motivational factors. On a five-point Likert scale from 1 (strongly disagree) to 5 (strongly agree) participants were asked to rate their current level of i) expected challenge of a task ("This task is a real challenge for me"); ii) probability of success ("I think I am up to the difficulty of this task"); iii) participants' interest ("I would work on this task even in my free time"), iv) anxiety of failure ("I'm afraid I will make a fool out of myself"). Our participants answered the QCM via the online survey provider Soscisurvey for every single HRE task. For further analysis, we computed means of the four sub factors after inverting items that were pooled differently.

Measures on Cognitive Load

As commonly applied in current research on Cognitive Load (Schmeck, Opfermann, Van Gog, Paas, & Leutner, 2015), we integrated the Perceived Task Difficulty Scale (Bratfisch, Borg, & Dornic, 1972), and the Mental Effort Scale (Paas, 1992) in both studies. The Cognitive Load Scales were included to measure if the students recognized the increasing complexity of the HRE tasks, and to rate how high or low their mental effort during the HRE tasks was. Participants rated their Perceived Task Difficulty on a 7-point Likert Scale, ranging from 1 (very very easy) to 7 (very very difficult). Additionally, students were asked to rate their invested amount of mental effort on a 7-point Likert Scale, ranging from 1 (very very low) to 7 (very very high) via the online survey provider Soscisurvey. We computed the means of each scale for further analysis.

Study Procedure

We conducted both studies with a quasi-experimental setting and a within-subject design during the winter term 2018/2019 at one German and one North American university. Both studies were included in the practical phase of the HRE course. After providing written informed consent, the participants received a randomly-assigned pseudonym. In the beginning of both studies, participants were asked to answer an online questionnaire on socio-demographics. For the four HRE tasks, we applied a similar procedure to collect data. After participants had read the assignment of the current HRE task, they were asked to rate their current level of motivation (QCM) regarding the HRE task at hand. After they had finished the current HRE task, they answered the two Cognitive Load Scales on Mental Effort and Perceived Task Difficulty.

3.5 Results

Results for RQ 1: Does students' performance in solving HRE tasks improve with increasing experience in HRE?

In order to answer RQ 1, we conducted a repeated-measures ANOVA of solution probabilities and solution times with the software IBM SPSS Statistics (please note that all following calculations were performed with this software, which is not explicitly mentioned again). As all participants had the same study procedure, we were able to merge the participants from both samples in our analysis. The repeated-measures ANOVA for comparing the mean of solution probabilities across the four HRE tasks revealed that students' solution probability decreased significantly in the most complex HRE task 4, *F* (3,35) =7.09, *p* =.00, η^2 = .38. It should be noted, however, that the mean solution probability (*M* = 77.2, *SD* = 31.4) was still at a satisfactory level. The results of a repeated-measure ANOVA showed significant differences between the means of solution time across the four HRE tasks, with *F* (3; 17) = 5:66, *p* = .03, η^2 = .50. The post-hoc analysis revealed that the solution time differed significantly between all tasks, except for solution time between tasks 1 and 3, and tasks 2 and 4. Table 3 summarizes descriptive data of solution probabilities and solution times.

Table 3. Means (*M*) and standard deviations (*SD*) of solution probabilities (in %; N = 38) and of solution times (in hh:mm; N = 20).

	Solution Probability (%)		Solution Time	
	M	SD	*M*	*SD*
HRE Task 1	97	5.9	03:19	02:39
HRE Task 2	95	16.5	06:31	04:45
HRE Task 3	93	14.8	03:03	02:34
HRE Task 4	77	31.4	05:07	04:05

Results for RQ 2: How does the HRE course affect the students' levels of motivation over all HRE tasks?

In order to analyze if the practical phase of our course design supported continuous motivation (RQ 2), we computed the repeated-measure ANOVA of the QCM. The analysis revealed no significant differences for the students' levels of motivation across the four times of measures. Within the dissertation, I added repeated-measure ANOVA's to test for differences in the four motivational factors Interest, Probability of Success, Anxiety, and Challenge over the four HRE tasks. The results showed no significant differences between the participants' levels of Interest, with F (3, 35) = 2.09, p = .11, η^2 = .15; for Probability of success with F (3, 35) = .73, p = .53, η^2 = .05; and Anxiety with F (3, 35) = .16, p = .92, η^2 = .14. We found a significant difference in Challenge with F (3, 35) = 5.29, p = .00, η^2 = .31. Post-hoc analysis revealed significant differences in individuals' levels of Challenge between task 1 and 4 (p = .03); 2 and 4 (p = .04); 3 and 4 (p = .00)., showing that students perceived the challenge of task 4 as significantly higher than the challenge of the other HRE tasks. The descriptive analyses showed medium to above-average levels of motivation over all HRE tasks (Table 4).

Table 4. Descriptive data with means (*M*) and standard deviations (*SD*) of the four QCM-factors (N = 38) with (1=strongly disagree; 5 = strongly agree).

	HRE Task							
	1		2		*3*		*4*	
QCM Factor	M	SD	M	SD	*M*	SD	*M*	SD
Interest	3.31	0.58	3.24	0.61	3.11	0.74	3.52	0.79
Challenge	3.27	0.78	3.25	0.89	3.00	0.91	3.66	0.74
Probability of Success	2.64	0.47	2.58	0.40	2.63	2.51	2.73	0.45
Anxiety	2.55	0.73	2.60	0.80	2.52	0.69	2.65	0.90

Results for RQ3: Do students perceive the growing complexity of the four HRE tasks?

In order to analyze if students perceived the increasing complexity of the single tasks, we conducted a repeated-measures ANOVA for the two Cognitive Load scales. The analysis revealed that students reported a significant higher Mental Effort in HRE task 4 compared to task 1 with, *F* (3, 35) = 3.56, *p* = .024, η^2 = .23. The repeated measures ANOVA for the Perceived Task Difficulty scale showed significant results between the means of task 2 compared to task 3, and compared to task 4, with *F* (3, 35) = 18.77, *p* = .000, η^2 = .62. These results showed that students perceived the growing complexity of the tasks. I added a summary of the descriptive data of participants' cognitive load in Table 5.

Table 5. Descriptive data with means (*M*) and standard deviations (*SD*) of mental effort and perceived task difficulty in the four HRE tasks with N=38 (1= very very low / high; 7 =very very easy / difficult).

	HRE Tasks							
	1		2		*3*		*4*	
Variable	M	SD	M	SD	*M*	SD	*M*	SD
Mental Effort	3.45	1.43	4.42	1.30	4.42	1.17	4.26	1.59
Task Difficulty	3.67	1.24	3.61	1.29	5.39	1.26	5.24	1.71

3.6 Discussion

The contribution of this chapter is twofold. First, we derived concrete guidelines for an HRE course structure based on prior results from educational and psychological research. The goal was to design the course to promote HRE skill acquisition in students with relevant backgrounds, for example in cyber security or electrical engineering. Second, we evaluated in two exploratory studies if the HRE course promoted HRE skill acquisition in students. Our results suggested that our HRE course may be a suitable instrument to prepare students to participating in our study on HRE problem solving.

During HRE, hardware reverse engineers analyze both graphical and textual representations of a gate-level netlist. Consequently, a course that aims to prepare students in solving realistic HRE tasks, should support students in working with and learning from those multiple HRE representations. As there was a complete lack of educational HRE courses in university programs, we developed the course based on prior findings from educational and psychological research. Against the background of these prior research findings we derived concrete guidelines for the HRE course. Those guidelines suggested to include specific instructional principles that should support students in acquiring specific competencies (e.g., perceptual competencies) that in turn would prepare them in working with and learning from multiple representations during HRE problem solving. Furthermore, following the theoretical considerations of the ACT-R model (Anderson 1982), we divided the course into two phases. The goal of the lecture phase (i.e., first phase of the course) was to teach students relevant declarative HRE knowledge (e.g., Boolean algebra; microchip architectures). The second phase, the practical phase, consisted of four practical exercises in which students were asked to solve HRE tasks in a realistic setting (i.e., applying the HRE tool HAL; solving realistic HRE sub-tasks). In order to solve those practical tasks, students needed to apply HRE knowledge, they had previously acquired in the lecture phase. By actively applying the declarative knowledge to solve the practical HRE tasks, the declarative knowledge should be gradually transformed into skills (procedural knowledge).

In the second part of this chapter, we evaluated if our course promoted the acquisition of HRE skills in students with relevant backgrounds, and analyzed their problem-solving performance in the four HRE tasks of the practical phase. Our results showed that students achieved high solution probabilities in HRE tasks 1, 2 and 3. Thus, we hypothesized that the lecture phase of the HRE course enabled the acquisition of HRE skills that supported the students to solve the HRE tasks. As the first three HRE tasks differed in their reversing goals but were developed based on the same cipher, we hypothesized that students were able to apply knowledge and skills they acquired in the analysis of HRE task 1 in order to solve HRE tasks 2 and 3 with high solution probabilities. Furthermore, our analysis revealed that students spent more time for solving HRE task 2 than for solving task 1. As students were asked to implement automated solutions in task 2 for the first time, the longer solution times in task 2 were not surprising. In terms of solution times, we found that the students tended to complete task 3 faster than task 2 (although task 3 was more complex than task 2). We hypothesized that memory and fluency-building processes were involved, and students developed their solution based on declarative and procedural HRE knowledge (Koedinger et al., 2012). In other words, students applied their knowledge gained in task 2, for solving task 3 faster, thus demonstrating that they built a set of HRE knowledge and skills (including sets of problem-solving strategies) that enabled them to become more fluent.

Furthermore, our results showed that the solution probability decreased in the most complex HRE task 4. This result may be based on the fact that HRE task 4 significantly differed from the previous HRE tasks in term of its complexity and requirements. However, we observed that the standard deviation was relatively high and that some students were able to achieve a very high solution quality in task 4 despite its complexity. This leads to two conclusions. On the one hand, weaker students should be supported in solving the more complex tasks (e.g., by offering support or by additional exercises that prepare them for the more complex HRE tasks). Second, this result also has consequences for the main study on exploring HRE problem solving. If the main study aims to investigate realistic problem-solving processes it might be a valuable approach to include only those students with the best solution quality in the complex HRE task. This would avoid that irrelevant influences (e.g. struggles with Python or HAL) could impair the analysis of HRE problem-solving process and bias the results.

Furthermore, we argue that the HRE course enabled students to solve the representation dilemma (Rau, 2017) as the participants achieved high solution probabilities in the HRE tasks. As proposed by our established guidelines, we included specific instructional (e.g., self-explanation; Koedinger et al.; 2012) and task principles to promote the acquisition of conceptual and perceptional competencies as well as connection-making abilities (Rau, 2017). We emphasize that future HRE courses should include those principles. Based on the established guidelines, we included spacing and testing (Pashler et al., 2007) as well as worked examples (Sweller & Cooper, 2005) to support students' development of memory fluency-building processes and induction and refinement-processes (Koedinger et al., 2012). We hypothesize that these principles supported students HRE knowledge compilation (Anderson, 1987) and thus, may have enabled them to acquire HRE skills. These two instructional principles should be part of future HRE courses.

According to Litzinger and colleagues (2011) we aimed to achieve higher levels of motivation that may lead to greater cognitive engagement and learning. Following our established guidelines, we included realistic HRE tasks and the HRE tool HAL. Our analysis revealed that students had above-average levels of motivation throughout the four HRE tasks. We assume that the integration of stimulating and realistic HRE tasks on an appropriate difficulty level (growing complexity) that had to be solved with the realistic HRE tool HAL led to the students' high levels of motivation. Therefore, those realistic aspects of the lecture phase should still be included in future HRE courses.

3.7 Limitations and Future Work

This presented work has limitations that should be investigated. This work is limited by the small sample sizes. Future studies on HRE skill acquisition should conduct analysis with larger sample sizes. Furthermore, our study did not specifically evaluate if the included instructional principles significantly influenced the acquisition of perceptual and conceptual competencies. We can only hypothesize that the included principles (e.g., worked examples) are suitable for acquiring HRE skills, but not if they are the optimum for supporting HRE skill acquisition. It would be preferable in future studies to quantify the influence of specific instructional principles and if they support the acquisition of competencies during HRE skill acquisition. Furthermore, our data led us to assume that declarative knowledge had been transformed into procedural knowledge (skills). A closer examination of these two knowledge types assigned to the two phases of the HRE course might prove interesting. Moreover, this study is a first investigation to fill the research gap of skill acquisition in HRE, leading to the fact that the generalizability to other areas is limited. Our research prompted us to make several observations about potential future research on HRE skill acquisition. In order to analyze problem-solving processes in HRE, future studies should focus on occurred errors and difficulties during HRE. Finally, the integration of a second complex task would help reveal more individual differences between students.

3.8 Conclusion

Researchers who aim to explore cognitive processes in HRE face the methodological problem that HRE experts are unavailable for research. One methodological approach that could enable such studies is to involve trained students. So far, no university-level HRE course existed and HRE training happened almost on the job. Furthermore, there was an almost complete lack of research that has explored how HRE skills and knowledge could be acquired.

Against the background of prior psychological and educational research findings, we derived specific guidelines for a course design aiming to promote HRE skill acquisition in students with relevant backgrounds. Within the scope of two exploratory studies, we evaluated the HRE course and its effectiveness in teaching HRE skills and knowledge. Our results showed that students (enrolled in Bachelor's and Master's cyber security and electrical engineering programs) were able to acquire HRE knowledge and skills. The students achieved very high to high solution probabilities in specific HRE tasks. In terms of future runs of the HRE course, educators need to consider that in the most complex task 4, some students (e.g., with lower scores in processing speed) achieved lower solution probabilities. Thus, future HRE courses should include specific exercises how to process and work with visual information received from the graphical representation of a gate-level netlist to support weaker students.

In summary, we hypothesize that the developed HRE course may be a suitable instrument to promote HRE skill acquisition in students with relevant backgrounds. Therefore, we suggest to include this HRE course as an HRE training to prepare students to solve a realistic HRE task of the main study.

4 Problem Solving in Hardware Reverse Engineering

Disclaimer: *The content of this chapter was previously published and submitted as parts of two papers. The introduction, methods, results, and general discussion concerning HRE problem-solving processes and correlations with levels of expertise were taken from the paper "The Anatomy of a Hardware Reverse Engineering Attack: Insights into Cognitive Processes during Problem Solving" that is currently under review at the journal ACM TOCHI. This paper was written together with my co-authors Steffen Becker, René Walendy, Christof Paar, and Nikol Rummel.*

Furthermore, parts of this chapter concerning the correlation of cognitive abilities with HRE problem solving (methods, results, discussion) were taken from the published paper "An Exploratory Study of Hardware Reverse Engineering – Technical and Cognitive Processes" (Becker et al., 2020) that was presented at the 16th Symposium on Usable Privacy and Security (SOUPS) in August 2020 together with my co-authors Steffen Becker, Nils Albartus, Nikol Rummel, and Christof Paar. Please note, that some sub-parts of the papers were not relevant for the overall argumentation of the dissertation, and were not included.

As both papers were conducted together with my co-authors, I will use the academic "we" to highlight this fact.

4.1 Introduction and Contributions

As outlined in greater detail above (see Background), HRE is a common tool to retrieve crucial information from an unknown chip design. As fully-automated HRE tools do not yet exist, the analysts' cognitive processes and cognitive abilities are hypothesized to mainly form the primary determinants of success (e.g., Becker et al., 2020). Nevertheless, the analysis of those underlying cognitive processes and factors in HRE has thus far been limited to a small amount of prior research. Lee and Johnson-Laird (2013) were one of the first to define reverse engineering of Boolean systems as a specific type of human problem solving. Besides these first results, HRE problem-solving processes and especially their time-efficiency or the ways in that they are impacted by levels of expertise or cognitive abilities, remain so far poorly understood.

We pursue those research gaps by systematically analyzing the problem-solving processes of hardware reverse engineers who solve a realistic HRE task. Thereby, we contributed to the overarching research goal (see Research Goals) by analyzing underlying problem-solving processes in HRE.

Furthermore, prior research on HRE problem solving is also lacking in analyzing the influence of expertise or of cognitive abilities on the problem-solving performance. Therefore, we included participants with different levels of HRE expertise and analyzed if their levels of expertise may lead to differences in the problem-solving process. Thereby, we contributed the sub-goals 1 and 2 of this thesis (see Research Goals). In the main study, we included experienced students enrolled in cyber security and electrical engineering programs. Beforehand, they acquired a sufficient amount of HRE knowledge and skills by successfully passing the HRE training as it has been demonstrated to prepare students with relevant backgrounds to solve realistic HRE tasks (see Chapter 3). Prior research on HRE problem solving is also strongly limited in exploring the role of an individual's level of intelligence on the HRE problem-solving performance. Hence, we analyzed participants on different levels of intelligence and sub-factors of intelligence to retrieve insights on the role of intelligence in HRE problem solving.

Against this background, we formulated the following research questions (RQs):

I. RQ1a. Which problem-solving processes can be observed while hardware reverse engineers are solving a realistic HRE task?
II. RQ1b. Are there differences in the HRE problem-solving process between analysts with different levels of expertise?
III. RQ2. Can differences in the time-efficiency of applied problem-solving strategies be identified?
IV. RQ3. Do cognitive abilities play a role in HRE problem solving?

We investigated these RQs by systematically analyzing the problem-solving processes of hardware reverse engineers on different levels of expertise and cognitive abilities who were asked to solve a realistic HRE task. To that end, we conducted an empirical study with nine hardware reverse engineers (eight top-performing intermediates; one HRE expert). In order to systematically explore applied problem-solving strategies and their time-efficiency, we analyzed 2445 single log entries by applying an iterative open coding scheme. Furthermore, we pursued how the level of expertise and cognitive abilities was correlated to the problem-solving performance.

In summary the contributions of this chapter are:

1) Based on our qualitative log-file analysis by applying an iterative open coding based on Grounded Theory (Strauss & Corbin, 1998), we developed a detailed hierarchical HRE problem-solving model. This model consisted of and defined 103 discrete problem-solving actions that hardware reverse engineers applied to solve the HRE task.
2) We provided unique and in-depth insights into applied problem-solving strategies and their time-efficiency.
3) We presented differences between the study participants with varying levels of expertise, as two intermediates were able to achieve time-efficient solutions that were comparable to the HRE expert.
4) Our results suggested that besides expertise, the intelligence sub-factor working memory may play a role in time-efficiently solving the realistic HRE task.

4.2 Methods

Study Overview

We conducted the empirical study with nine reverse engineers with intermediate and expert levels of expertise during the summer term of 2019 at a German university. The participants on intermediate levels acquired a sufficient amount of HRE skills and knowledge by successfully completing the HRE training (see Chapter 3). We collected behavioral log file data that consisted of 1141 executed scripts, 68 console inputs, and 1217 manual activities from nine participants. The study participants successfully solved a realistic HRE task over the period of two weeks with the HRE tool HAL. In addition, participants were asked to answer a series of additional questionnaires (e.g., self-assessment of domain-specific expertise; intelligence test). In order to systematically examine applied problem-solving strategies, we applied a well-established iterative open coding methodology based on the Grounded Theory approach by Strauss and Corbin (1998). Furthermore, we qualitatively related the identified problem-solving strategies to time on task as well as to levels of expertise and cognitive abilities.

Participants

In summary, eight students on intermediate levels in HRE and one HRE expert voluntarily participated in our study. A total of 22 students who were enrolled in either their last year of a three-year Bachelor's cybersecurity program or in a Master's cybersecurity program composed the population of the HRE intermediates. In order to achieve an intermediate level of HRE expertise that enabled them to solve the realistic HRE task, all students had successfully completed the HRE training (see Chapter 3). Of the original 22 students, eight withdrew their participation in the study or could not be included in the analysis due to missing or incomplete data.

Eight top-performing students (mean age *M* = 24 years; *SD* = 4 years) were selected from the group of intermediates. This was motivated by the fact that we aimed to derive a representative and generalizable HRE problem-solving model that would represent realistic behaviors of hardware reverse engineers as closely as possible. Those eight top-performing students seemed to be the most suitable proxy for realistically analyzing hardware reverse engineers, besides the HRE expert. Furthermore, by selecting these top-performing students, we aimed to avoid the inclusion of HRE-unrelated issues in our HRE problem-solving model, such as insufficient programming skills or difficulties in the application of the HRE tool HAL (see Description of HAL). This top-performing group of intermediates achieved a mean solution percentage of 97.5% with 7.1% standard deviation over all four HRE training tasks. The HRE task was solved by the top-performing intermediates with a mean solution percentage of 98.5% with a standard deviation of 1.9%. The intermediates' solutions were graded collaboratively by three teaching assistants based on a detailed gradebook with sample solutions. The teaching assistants assigned solution probabilities on a scale ranging from 0% to 100%. The solution probability of 100% was assigned to solutions in that participants completely removed the implemented watermark from the netlist (see description of watermark task).

As previously outlined (see Background), recruiting HRE experts can pose a methodological challenge for researchers. We were able to recruit at least one HRE expert by leveraging the professional network of one of the authors. This HRE expert was a researcher in the field of hardware security and stated to have five years of experience in conducting HRE. Furthermore, the expert claimed to typically spent between 20 and 30 hours per week on solving HRE tasks. In addition, the expert invested between 20 to 30 hours per week in HRE-related tasks such as software and hardware programming activities. Finally, the expert was characterized by considerable amount of prior domain-specific experience in HRE-related topics (e.g., high-level and low-level programming languages), and by a significant amount of domain-specific knowledge in HRE-related topics such as chip architectures (e.g., FPGA) or crucial netlist components (e.g., FSM).

Ethical Considerations

All participants (the 22 intermediates and the expert) provided written informed consent before entering the study. We emphasized that the study participation was voluntary and outlined that a withdrawal from the study was possible at any time and without stating reasons (including the deletion of all study-related data). For the completion of study-related materials and tasks, the 22 students received a monetary compensation. In order to protect participants' privacy, we randomly assigned pseudonyms to the participants that were used instead of their clear names on all study-related materials and during all study-related activities. The pseudonyms were subsequently replaced with numbers as part of the data analysis. Since there was no ethics committee at the institute at the time the study was conducted, the data protection officer of the German university at which the study was conducted, reviewed and approved the privacy-protection procedures of this study.

Materials

HRE Skill Training for Students

In order to support students in solving the realistic HRE task, it was essential that those students acquired a suitable amount of HRE-specific knowledge and skills beforehand. Therefore, we included the 14-week HRE training course that was proven efficient in transforming declarative HRE knowledge into HRE skills (see Chapter 3). Even if not yet HRE experts, the high solution percentages in the four HRE training tasks of the eight top-performing students proved that they had acquired an intermediate level of expertise through the successful completion of the HRE course. Despite the fact that the four HRE training tasks were built upon real-world HRE scenarios (e.g., extraction of an AES; reversing an obfuscated circuit), the HRE training course did not include any best practice or solution strategies specific to the realistic HRE task of the study. This was an effort to avoid biasing or artificially enhancing the performance of the participating students in the realistic HRE task. At the same time, we feel that the missing preparation of the students on how to specifically solve the realistic HRE task during the HRE training did not deprive them of essential prior experience. We would like to emphasize that the HRE course was holistic and comprehensive, and that no training includes every possible novel scenario that may have to be solved in the specific field. As the HRE expert already possessed the required HRE knowledge and skills, and also collected prior experience in working with the HRE tool HAL, the expert did not complete the HRE training.

HRE Task

In order to capture and observe realistic problem-solving processes, it is essential include a representative task that is directly sampled from a real-world situation (Charness & Tuffiash, 2008). As a well-known and widely used method, so-called watermarks, are applied to detect hardware counterfeits (Abdel-Hamid, Tahar, & Aboulhamid, 2003). Hardware manufacturers mark their IP by embedding those watermarks in their microchips to impede theft of their innovative developments. Accordingly, the detection of the watermark in an unauthorized clone of the original netlist can prove that the netlist, and thus the IP, has been counterfeited (Becker et al., 2020). Consequently, when an analyst plans to build an illegal clone of the original netlist, the watermark has to be detected and removed from the netlist.

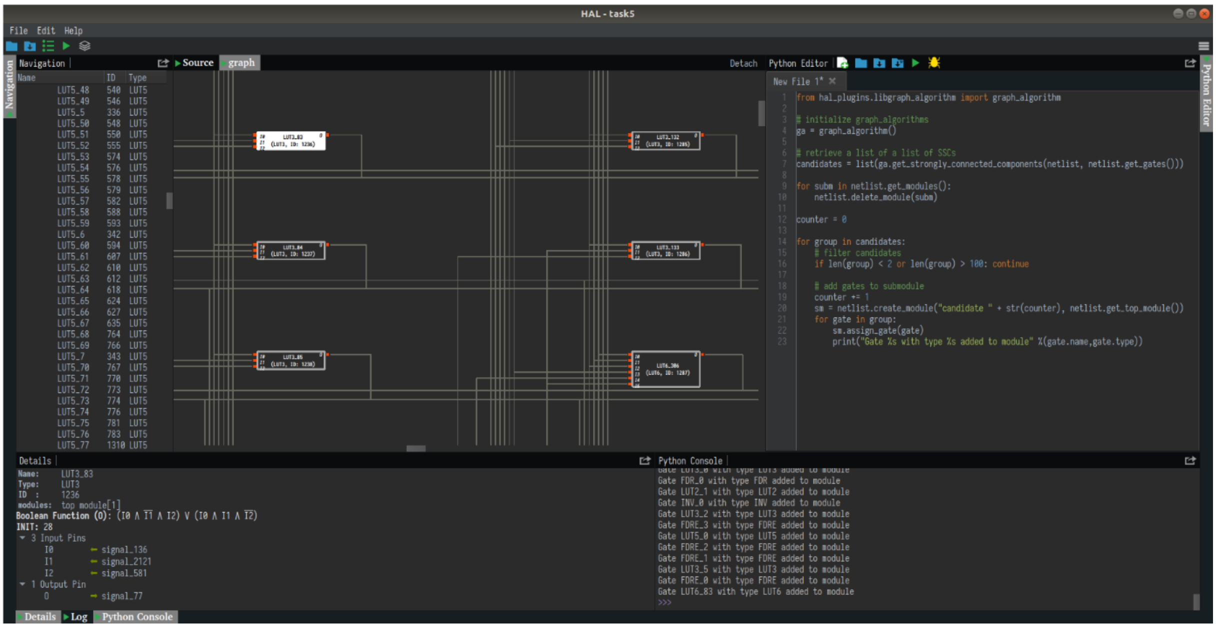
Against this background, the HRE task of the study directed the study participants to identify and remove a watermark from the gate-level netlist – a challenge that can be found in real-world scenarios as shown above. The gate-level netlist of the HRE task consisted of 4.653 Boolean gates, memory components, and their interconnections and contained a watermark scheme as proposed by Schmid, Ziener and Teich (2008) implementing a copyright protection. The goals of the HRE task were to remove the watermark and to clone the circuit without the watermark whereby the infringement of the IP would be unable to be proven. In order to remove the watermark, participants were asked to identify the specific netlist components that implemented the watermark in a first step. In the second step, participants were asked to develop a custom technique to extract and remove the watermark from all the identified components.

HRE Tool HAL

HAL was developed by Fyrbiak and colleagues (2018a) and was included in our study as a tool that enabled participants to conduct HRE analysis in order to remove the watermark from the given gate-level netlist. At the same time, HAL also assisted the researchers in the analysis of problem-solving processes, as the tool enabled the automatic recordings of behavioral log files of every single participant.

Researchers and experts in the HRE domain use HAL (available on GitHub, Chair for Embedded Security, 2019; Wallat et al., 2019) as a state-of-the art HRE tool to analyze netlists of unknown chip designs. HAL provides a rich GUI to the analysts (Figure 3), and allows reverse engineers to focus on conducting HRE analysis with no need for further tool development. Nevertheless, HAL does not provide any (semi-)automated netlist analysis methods. Both, manual / visual and script-based analysis are possible due to HAL's GUI. Hardware reverse engineers are supported in their manual analysis of specific netlist components such as gates and their interconnections by a textual and graphical representation of the netlist, which is integrated in HAL's GUI. Furthermore, as it is a common method in HRE practice and also part of the complex HRE task, HAL allows analysts to interactively script and test their reversing methods by in integrated Python shell. Therefore, HAL supports the script-based analyses of and the interaction with the netlist by providing multiple reverse engineering-specific Python commands.

Students who participated in our study, practiced how to use both manual and script-based analysis with HAL during the four HRE training tasks of the HRE course. The HRE expert gained prior experience in the expert's daily work practice that included netlist analysis with HAL. In order to simulate a realistic HRE scenario within our study that was comparable to that in which HRE experts normally work, we included the following aspects: a manual for HAL that contained i) specific operating instructions, ii) detailed information of HRE-specific Python commands, iii) examples of code snippets demonstrating HAL' capabilities.



Name	ID	Type
LUT5_48	540	LUT5
LUT5_49	546	LUT5
LUT5_5	336	LUT5
LUT5_50	548	LUT5
LUT5_51	550	LUT5
LUT5_52	555	LUT5
LUT5_53	574	LUT5
LUT5_54	576	LUT5
LUT5_55	578	LUT5
LUT5_56	579	LUT5
LUT5_57	582	LUT5
LUT5_58	588	LUT5
LUT5_59	593	LUT5
LUT5_6	342	LUT5
LUT5_60	594	LUT5
LUT5_61	607	LUT5
LUT5_62	610	LUT5
LUT5_63	612	LUT5
LUT5_64	618	LUT5
LUT5_65	624	LUT5
LUT5_66	627	LUT5
LUT5_67	635	LUT5
LUT5_68	764	LUT5
LUT5_69	766	LUT5
LUT5_7	343	LUT5
LUT5_70	767	LUT5
LUT5_71	770	LUT5
LUT5_72	773	LUT5
LUT5_73	774	LUT5
LUT5_74	776	LUT5
LUT5_75	781	LUT5
LUT5_76	783	LUT5
LUT5_77	1310	LUT5

```python
from hal_plugins.libgraph_algorithm import graph_algorithm

# initialize graph_algorithms
ga = graph_algorithm()

# retrieve a list of a list of SSCs
candidates = list(ga.get_strongly_connected_components(netlist, netlist.get_gates()))

for subm in netlist.get_modules():
    netlist.delete_module(subm)

counter = 0

for group in candidates:
    # filter candidates
    if len(group) < 2 or len(group) > 100: continue

    # add gates to submodule
    counter += 1
    sm = netlist.create_module("candidate " + str(counter), netlist.get_top_module())
    for gate in group:
        sm.assign_gate(gate)
        print("Gate %s with type %s added to module" %(gate.name,gate.type))
```

Name: LUT3_83
Type: LUT3
ID : 1236
modules: top module[1]
Boolean Function (O): $(I0 \land \overline{I1} \land I2) \lor (I0 \land I1 \land \overline{I2})$
INIT: 28

```
▾ 3 Input Pins
    I0    ← signal_136
    I1    ← signal_2121
    I2    ← signal_581
▾ 1 Output Pin
    O     → signal_77
```

```
Gate LUT3_6 with type LUT3 added to module
Gate FDR_0 with type FDR added to module
Gate LUT2_1 with type LUT2 added to module
Gate INV_0 with type INV added to module
Gate LUT3_2 with type LUT3 added to module
Gate FDRE_3 with type FDRE added to module
Gate LUT5_0 with type LUT5 added to module
Gate FDRE_2 with type FDRE added to module
Gate FDRE_1 with type FDRE added to module
Gate LUT3_5 with type LUT3 added to module
Gate FDRE_0 with type FDRE added to module
Gate LUT6_83 with type LUT6 added to module
>>>
```

Figure 3. Screenshot of HAL's GUI. Left: Text-based representation of netlist components. Middle: Graphical representation of the unknown netlist. Right: Python editor for script-based netlist interactions. Bottom left: Details widget providing additional information

Collected Data

Demographic Questions

All participants were asked to provide information about their sociodemographic backgrounds. Therefore, we included two short questionnaires in the study. One of these questionnaires was developed for the student participants, and asked them to answer questions about their age, major, and target degree. The second questionnaire was designed for the HRE expert and asked the expert to self-rate the level of HRE expertise including questions on age, highest level of education, current job position, HRE expertise level, year of relevant experience in HRE, as well as hours spent on performing HRE per week. In developing the expert questionnaire, we were oriented to the measurement of the level expertise among software reverse engineering experts according to Votipka and colleagues (2020).

Cognitive Abilities

We included the Wechsler Adult Intelligence Scale (WAIS-IV) to assess participants' cognitive abilities (Wechsler, 2008). The WAIS-IV included the measurement of three sub-factors of intelligence. First, the ability to accurately interpret and work with visual information was measured by the factor perceptual reasoning, that consisted of three tests: Block design (participants were asked to rearrange 3-dimensional blocks to match patterns), matrix reasoning (participants completed 2-dimensional series of figures), and visual puzzles (participants chose three figures out of six to build a 2-dimensional geometric shape). The second intelligence sub-factor working memory reflected the ability to memorize information and to perform mental operations by applying that information. It consisted of two tests: Digit span (participants repeated a series of numbers spoken to them), and arithmetic (participants solved under time pressure several arithmetical problems spoken to them). Processing speed measured the participants' ability to quickly and efficiently process visual information. It consisted of two tests: Symbol search (participants were asked to search symbols accurately in a given time limit), and coding (participants needed to transcribe a unique geometric symbol with its corresponding Arabic number accurately under time pressure).

Behavioral Log Files

The HRE tool HAL automatically generated and saved log files from each of the nine participant's problem-solving steps during the netlist analysis. These nine log files consisted of 148 to 467 single log entries. Log entries included text files with time-stamps and one of the following events:

i. A script-based analysis step with the executed Python script, information about the syntactical correctness of the Python script, and the corresponding console output;
ii. A short Python console input with information about syntactical correctness of the Python script, and the corresponding console output;
iii. A manual analysis step (e.g., selection of a netlist component such as a gate or a net) and the unique identifier of the selected netlist component;
iv. Marker for (in)activity phases including information about the duration of these phases.

In summary, the nine log files of the hardware reverse engineers included 2445 single events (of which 1141 executed Python scripts, 68 console inputs, 1217 manual netlist component selections, and 19 idle events). In order to prepare for the subsequent open coding analysis, the log files were pre-processed by displaying the events together with associated information in tabular form. Figure 4 shows a section of such a pre-processed log file.

Timestamps	Type	Attribute	Component ID or File Name
…	…	…	…
3,822 s	Script	Working	003822_Participant8.py
3,856 s	Manual	Gate	514
4,240 s	Script	Erroneous	004240_Participant8.py
4,268 s	Script	Working	004268_Participant8.py
4,334 s	Manual	Gate	531
4,341 s	Manual	Net	2,437
…	…	…	…

Figure 4. Example Section of Participant 8's Pre-Processed Log File with 6 of 153 Total Events (Extract of Script of Participant 8 is Attached in the appendix).

Data Analysis Method

Iterative Open Coding based upon Grounded Theory

Each event of the nine participants was qualitatively analyzed by applying an iterative open coding approach that was based on the Grounded Theory methodology by Strauss and Corbin (1998). Known as a well-established and standard research method from the social sciences, the Grounded Theory methodology is often applied in research domains in which theories and models are lacking thus far (Charmaz & Belgrave, 2007). Since this is also the case for theories and models of cognitive processes in HRE and Grounded Theory provides explicit guidelines that facilitate the analysis of HRE problem-solving processes, we decided to apply such an iterative open coding.

The process of the iterative open coding was as follows. As some single consecutive events overlapped strongly in terms of content, we decided to group them first. This combined 1232 consecutive events into segments that should facilitate the qualitative analysis. These segments consisted mostly of manual netlist interactions and barely of consecutive console inputs or executed Python scripts. Before we could assign open codes, each single event had to be described in detail. This very detailed description formed the basis for developing an open coding scheme that could then be iteratively refined. The annotation of each event and segment consisted of the following aspects:

I. The recorded problem-solving step was described (e.g., "The executed script iterates over all netlist components and checks if their name contains the string Look-up Table (LUT).", or "Manual selection of a netlist component implementing a watermark.").

II. The duration (in seconds) of the problem-solving step was added.

III. Any changes compared to the previous step were described (e.g., “The participant added three lines of code containing one print statement, one if-clause, and the reversing-specific Python function *get_data_by_key()”*).

IV. The observed behavior was explained (e.g., “The watermarking is implemented by LUTs”. Therefore, the participant filters the netlist components for LUTs.”).

We assigned one or more open codes that captured the most relevant annotations. These open codes were assigned to the single segments in the evaluation table. Examples for assigned open codes are “script-based inspection of watermark candidates” or “successful correction of syntactical errors”. The iterative open coding procedure started with two researchers who collaboratively annotated and encoded the log files of four participants with the goal to create an initial code book. During this step, the initial open code book was continuously updated and previous segments were re-coded as necessary. Based on the initial code book, a third researcher annotated and encoded the log files of the five remaining participants. After all participants' problem solving had been encoded, all three researchers reviewed and discussed the annotations, include new codes in code book, and retroactively applied any such codes where applicable.

In summary, the final code book includes 103 different unique open codes that represent a compacted version of the single segment annotations and describe the observed HRE problem-solving processes of our nine participants. An external researcher independently coded 71 randomly selected segments from three randomly chosen participants with the existing code book and achieved an inter-coder reliability of 84.5%. Overall, we assigned the 103 unique open codes 1887 times across the 1232 annotated segments.

Taxonomy of Open Codes

Based on the final code book, we developed an HRE problem-solving taxonomy during several rounds of discussion. First, we grouped related open codes into problem-solving clusters. The grouping was based on similarities between single open codes. These problem-solving clusters were then subordinated into nine sub-categories. These sub-categories represented the participants' problem solving during the study (RQ1), and also build the foundation for the analysis of problem-solving strategies and their efficiency (RQ2).

Eight of the nine sub-categories could be arranged into two main categories. The first main category comprised all problem-solving processes that were related to Python programming actions. The second main category included all reversing-specific actions. Based on the HRE problem-solving taxonomy, we were able to develop a “language” to describe observed problem-solving processes and to analyze differences between participants (RQ1b).

Total Solution Time

Based on the time stamps in the automatically generated log files we were able to compute the total solution time per participant. During the log file analysis, we found idle events that were unrelated to the HRE task and could be clearly identified as breaks based on their annotations and wall-clock time (i.e., time of the day at which they occurred). Those idle times were excluded from the total solution time. We included the total solution time to identify differences in the time-efficiency of applied problem-solving strategies (RQ2).

Incorporating Open Code Duration

We noticed that the mere number of open codes did not provide a sufficient granular metric to describe the problem-solving strategies of the hardware reverse engineers (RQ2). Thus, we included the incorporation of participants’ solution time. Therefore, we first divided the total solution time of the participants in 50 time windows of 2% each[1].

We then analyzed which open code(s) were assigned in each of the 50 time windows. This resulted in a fine-grained measure of the relative amount of time that a participant spent on a single action (i.e., described by the annotated open code). Based on this metric, we were able to analyze which open codes were assigned towards the beginning, middle, or end of the participants’ time on task. In summary, the incorporation and detailed analysis of participants’ time spent on specific open codes, contributes to the analysis of applied problem-solving strategies and their efficiency to answer RQ2.

Summary of Study Procedures

Before the 14-week HRE training and the study started, all participants signed the informed consent document. Furthermore, all participants received a randomly assigned pseudonym they were asked to use instead of their clear names during the study. All participants answered the questionnaire on socio-demographics via an online survey provider. Attentionally, the intelligence test (i.e., WAIS-IV) was conducted in a 60 to 90 minutes face-to-face session with

[1] We also tested for smaller time windows, e.g., of 1%, but could not detect any gain in accuracy.

one of the researchers. After the successful completion of the HRE training, the HRE task materials consisting of a short task description, the watermarked netlist, and a copy of the paper in which the implemented watermark was introduced were distributed to the participants. The participants had two weeks to complete the HRE task, without precise specifications as to what time of day they wanted to work or whether they wanted to solve the task in one session or spread over several days. After completing the tasks, participants uploaded their log files to a server located at the university and received a monetary compensation for working on study-related tasks.

4.3 Results

In the following, the results of our analysis to answer the three RQs introduced in the beginning of this chapter (see Research Questionss) are presented. In order to answer RQ 1a, we developed an HRE problem-solving model based on a taxonomy were derived from the detailed analysis of the behavioral log files of the nine participants. In the context of RQ 1b, we build upon the foundational HRE problem-solving model to explore differences between the HRE expert's and the eight intermediates' problem-solving processes. Furthermore, we analyze time-efficiency of the applied problem-solving strategies to answer RQ 2. Finally, we explore the role of cognitive abilities in HRE problem solving (RQ3). We included a brief discussion after each of these three main results. All analyses were performed with self-developed analysis programs, as well as Microsoft Excel and MAXQDA.

4.3.1 Results RQ1a

RQ1a: Which problem-solving processes can be observed while participants are solving a realistic HRE task?

As previously described in the section Data Analysis Method (see Methods), we systematically analyzed and categorized the log files of the participants (eight intermediates, one HRE expert) after the completed the realistic HRE task of the study.

The 103 unique open codes were grouped into nine sub-categories that enabled us to describe the observed HRE problem-solving processes in detail and to develop an HRE problem-solving model, which is presented in Figure 5 below. Within our analysis, we found that eight of the nine sub-categories could be grouped into one of the two main HRE problem-solving categories: Reversing Actions or Code Development. The main category Reversing Actions combined all actions that directly related to the HRE problem-solving process. In contrast, the main category Code Development included actions that focused on the development of Python programming code. The ninth sub-category External Influences combined all actions that were not related to reverse engineering or coding activities and encompassed codes that included exogenous influences upon the HRE problem-solving process such as external interruptions (i.e., short breaks during the reversing process).

We assigned the open codes sorted by sub-categories to the individual actions of the nine participants. In the problem solving of P5 we did not identify any actions that could be assigned to the sub-category of External Influences. Table 6 provides an overview about the total number of assigned open codes per participant aggregated at the sub-category level.

Table 6. Code System with Numbers of Assigned Open Codes per Category, Sub-Category, and Participant. (A detailed view of all open codes and the frequency with which they were assigned to each participant is shown in the appendix.)

	P1	P2	P3	P4	P5	P6	P7	P8	Expert
	Reversing action								
Inspection and Information Gathering	10	17	15	11	23	17	25	19	23
Reversing Milestones and Sub-Goals	13	12	13	16	19	19	15	10	11
Reversing Problems	6	2	10	9	5	8	8	1	1
Reversing Strategy Decisions	11	17	21	24	31	17	41	9	37
Total	40	48	59	60	78	58	89	39	72
	Code Development								
Code Adjustment	49	39	42	49	70	38	41	20	63
Error Introduction	24	14	27	46	26	22	25	7	31
Test and Validation	27	12	30	32	45	14	22	7	43
Troubleshooting	63	24	49	74	52	39	35	20	49
Total	163	89	148	201	193	113	123	54	186
External Influence	23	7	5	9	0	3	19	8	3

Our developed HRE problem-solving model had a hierarchical structure that included four levels, progressing from general to specific: The highest level of the model included the main categories, the sub-categories were included at the second level, which were followed by clusters and finally by open codes at levels three and four. Figure 5 visualizes the HRE problem solving model with the first two levels including main categories, and sub-categories. The numbers of unique open codes specify how many different open codes were assigned to the sub-category or the cluster. The number of assigned open codes indicated how often a unique open code was assigned to a participant's problem-solving actions.

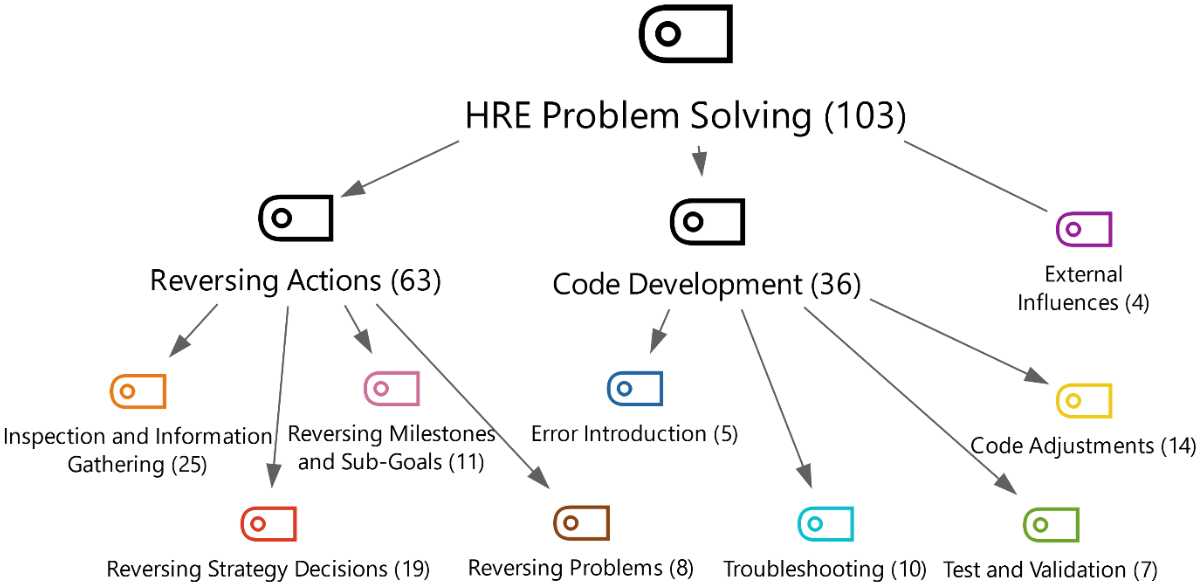


Figure 5. HRE Problem-Solving Model with the two main categories Reversing Actions and Code Development, and the nine sub-categories at level two. Numbers in brackets indicate the number of unique codes per sub-category.

Main Category: Reversing Actions

The main category Reversing Actions included 63 unique open codes that were assigned 543 times to actions of the nine study participants. The next analysis step revealed that these open codes belonged to one of the following four sub-categories: Inspection and Information Gathering, Reversing Strategy Decisions, Reversing Milestones and Sub-Steps, and Reversing Problems.

The first sub-category, Inspection and Information Gathering, included all actions that participants performed to retrieve information about the netlist and its components (e.g., assigned clusters Exploration or Identification) or actions that aimed to retrieve detailed information about (crucial) netlist components (e.g., assigned cluster Inspection). The number of open codes that was assigned to the cluster Inspection was larger than the number of open codes pertaining to the cluster Exploration or Identification. Within the sub-category Inspection and Information Gathering, manual netlist analysis steps were predominant (15 unique open codes; assigned 135 times), whereas the number of script-based netlist analysis actions was smaller (15 unique codes, assigned 25 times). The following open codes were assigned most frequently in this sub-category: in-depth manual inspection of watermark candidates (assigned 39 times), manual selection of irrelevant gates (assigned 14 times), and manual netlist exploration (assigned 13 times).

The second sub-category, Reversing Strategy Decisions, included actions or decisions that the participants performed to solve sub-problems of the HRE task. We organized the assigned open codes in this sub-category into the following three clusters: Strategies and Approaches (14 unique open codes, assigned 135 times), Change of Strategy (3 unique open codes, assigned 14 times), and Sub-Step Preparation (2 unique open codes; assigned 59 times). The most frequently assigned open codes in the sub-category Reversing Strategy Decisions were both open codes of the cluster Sub-Step Preparation and were assigned with equal frequency. The second most commonly assigned open codes in this sub-category were *Duplication of partial solutions for watermark candidates* (17 times) and *Reversion to a proven approach* (10 times).

The third sub-category, Reversing Milestones and Sub-Goals, contained open codes that described actions through which progress in the HRE problem-solving process was achieved. Examples for assigned open codes are *Identification of watermark candidates* or *Removal of the watermark*. This sub-category included the following clusters in descending order of significance: Achieving Milestones (4 unique codes, assigned 54 times), Achieving Sub-Goals (3 unique codes, assigned 21 times), Systematic Approach to Considering Milestones (3 unique codes, assigned 43 times).

The fourth sub-category, Reversing Problems, combined actions that indicated quite the opposite of the previous sub-categories. The codes grouped in the sub-category, Reversing Problems, contained problem-solving steps that were error-prone or that led to a dead end in the problem-solving process. We subdivided the sub-category in the following clusters: Confusion (3 unique codes, assigned 16 times), Failed Attempts (2 unique codes, assigned 12 times), and Lack of Understanding (3 unique codes, assigned 22 times). The most frequently assigned open codes in this sub-category were *Reversing-specific lack of understanding* (assigned 11 times), *Lost track of the reversing approach* (assigned 9 times), and *Dead end* (assigned 8 times).

Main Category: Code Development

The second main category, Code Development, grouped programming-related actions and included the following four sub-categories: Error Introduction, Troubleshooting, Test and Validation, and Code Adjustments. Every HRE task, including the HRE task, involved development of programming code or customized scripts as full automated HRE tool support did not exist. Therefore, the sub-category, Code Development, was considered in the analysis of HRE problem solving. Nevertheless, many open codes which were assigned to actions occurred in the context of the present HRE task, may also occur in contexts other than HRE.

The first sub-category, Error Introduction, included the following two clusters: Semantic Errors (2 unique open codes, assigned 98 times) and Syntactical Errors (2 unique open codes, assigned 115 times) that were identified during the development of Python code to solve the HRE task. For example, *Introduction of semantic errors* was an open code that was assigned 87 times in this sub-category.

The second sub-category, Troubleshooting, combined the actions grouped into two clusters: Error Search and Correction Attempts (5 unique open codes, assigned 136 times), and Error Correction (5 unique open codes, assigned 269 times). Examples for open codes of the second sub-category were *Successful correction of syntactical errors* (assigned 160 times), and *General debugging* (assigned 57 times).

The third sub-category, Test and Validation, combined two cluster: Focused program code testing and validation methods (5 unique open codes, assigned 181 times), and General program code testing and validation methods (2 unique codes, assigned 51 times). The sub-category, Test and Validations, included open codes such as *Targeted verification* (assigned 58 times) or *Manual netlist inspection for script validation* (assigned 38 times).

The fourth and final sub-category, Code Adjustments, involved actions of restructuring or simplifying programming scripts of document solutions. Allover, three clusters were developed: Creating Clarity (6 unique open codes, assigned 268 times), Cut, Copy and Paste (5 unique open codes, assigned 67 times), and Documentation (3 unique open codes, assigned 76 times). Open codes of this sub-category were, for example, *Improve clarity of console output* (assigned 109 times); *Reversion to previous code components* (assigned 38 times), or *Explanatory documentation* (assigned 31 times).

Sub-Category: External Influences

In addition to the sub-categories of the two main categories Reversing Actions and Code Development, our analysis revealed another sub-category, External Influences (4 unique open codes, assigned 74 times) that could not be classified under the two main categories. Open codes included in this sub-category were, for example, *External interruption* (assigned 35 times) or *Unintentional manual selection* (assigned 16 times). Although the behavioral observations summarized in this sub-category were not exclusive to HRE and could also be found in other programming-related tasks, it was very likely that they also occurred in real-world HRE scenarios and thus reasonably completed our HRE problem-solving model.

4.3.2 Discussion RQ1a

In order to systematically gather and analyze the behavioral log files of the nine participants, we applied an iterative open coding approach based upon the Grounded Theory method (Strauss & Corbin, 1998). Based on this analysis, we were able to develop a detailed and hierarchical model of HRE problem solving.

Our HRE problem-solving model included two main categories and nine sub-categories that were developed based upon 103 assigned open codes. While we do not deny the possibility that our model may be expanded by new codes that emerge through the analysis of a different task, we believe that our model covers all of the essential actions related to an HRE task at hand. Based on the resulting HRE model, we were able to conceptualize the observed HRE processes. This conceptualization supported us in generating a language that we applied to describe problem-solving strategies during HRE.

Summarized, we divided the above-presented model into four sub-categories that included reversing-specific actions with additional granularity at the cluster-level. Based on this, our model provided an in-depth perspective and enabled us to further explore fundamental expertise-related differences in participants' problem solving in our analysis of RQ1b and RQ2. The sub-categories were central components of the model as they represented the variety of approaches that the participants applied to solve the HRE task. Furthermore, the sub-categories indicated obstacles encountered during the problem-solving process. In order to identify expertise-related differences in the time-efficiency of participants' problem solving, we analyzed these sub-categories involving the time on task. Furthermore, we analyzed other actions that caused specific Reversing Problems or Reversing strategy decisions. We delved deeper into these sub-categories by answering RQ2.

4.3.3 Results RQ1b

RQ1b: Are there differences in the HRE problem-solving process between participants with different levels of expertise?

In the context of this RQ, we investigated whether there were differences in the problem-solving processes between the intermediates and the expert at the open-code level. Figure 6 shows the visualization of this comparison. The left side of the visualization included open codes that could only be observed in the expert's problem solving. All open codes of the expert came from the main category Reversing Actions. The right side of the figure showed the five most frequent open codes that could only be detected in the problem solving of the intermediates. Of these, three open codes came from the main category Reversing Actions and two from the sub-category External Influences. The middle part of the visualization showed the five most common open codes observed in both the intermediates and the expert’s problem solving. All five came from the main category Code Development. In summary, 52 unique open codes were assigned to the problem solving of the intermediates. Of these 52 unique codes, only 10 open codes fell into the main category of Code Development and 39 fell into the main category of Reversing Actions.

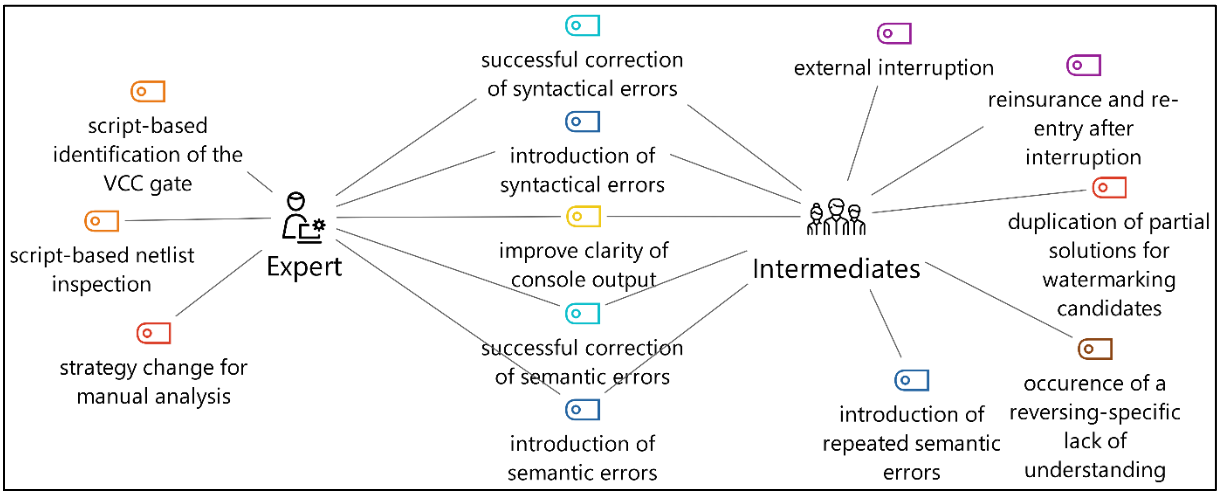

script-based identification of the VCC gate
script-based netlist inspection
strategy change for manual analysis
Expert
successful correction of syntactical errors
introduction of syntactical errors
improve clarity of console output
successful correction of semantic errors
introduction of semantic errors
Intermediates
external interruption
reinsurance and re-entry after interruption
duplication of partial solutions for watermarking candidates
occurence of a reversing-specific lack of understanding
introduction of repeated semantic errors

Figure 6. Comparison between intermediates and expert. Left: Most frequently observed codes of the expert. Middle: Most frequently observed codes shared by expert and intermediates. Right: Most frequently observed codes of the intermediates.

We found differences in the problem solving between the expert and the intermediates in the main category Code Development. Especially the distribution of the open codes *Introduction of repeated semantic errors* (assigned 11 times) and *Introduction of redundant code* (assigned 6 times) indicated a lack of Python-programming experience of some intermediates. In this context, we observed that the intermediates frequently used problem- solving actions associated with the open code *Anticipatory documentation* (assigned 8 times), serving to help the intermediates better plan and control their work steps. Otherwise, no differences were found between the expert and the intermediates in the distribution of unique open codes in the main category Code Development.

Differences in the main category Reversing Actions were mainly based on the open codes assigned to the intermediates. Unique approaches to problem solving by the intermediates were found in the Inspection and Information Gathering sub-category. These assigned open codes were mainly from the area of manual information gathering. On the other hand, two of the three open codes that could be assigned exclusively to the expert belonged to the area of script-based information gathering.

In the sub-category Reversing Strategy Decisions, we found that intermediates applied several diverse and unique processes as 13 open codes of this sub-category were exclusively assigned to intermediates. Furthermore, in the sub-category Reversing problems, our findings revealed that seven out of eight unique open codes were assigned only to intermediates. In terms of processes in the sub-category Reversing Milestones and Sub-Goals, seven of the eleven open codes were assigned to both the expert and the intermediates. Four codes in this sub-category could only be found in the HRE processes of the intermediates.

4.3.4 Discussion RQ1b

In the previous section, we conducted a comparison of the observed HRE problem-solving processes between the expert and eight intermediates at the open-code level. We found major differences between both the expert and the intermediates within the main-category Reversing Actions, mainly located in the sub-categories Inspection and Information Gathering, Reversing Strategy Decisions, and Reversing Problems.

Based on our findings on expertise-related differences in HRE that seemed to be mainly concentrated in the three previously mentioned sub-categories, we were able to draw a first precise picture of the applied problem-solving strategies. Therefore, the following analysis on time-based efficiency of applied problem-solving strategies (in the context of RQ2) focused on these three sub-categories. In order to analyze efficiency, we expanded our analysis of the number of assigned open codes by including a fine-grained metric based on the participants' time on task (see Methods).

As shown by the analysis, we found that the number of External Interruptions was higher for Intermediates than for the expert. We argue that intermediates may have tended to re-enter and re-orient themselves after experiencing external interruptions more frequently. Thus, we suggest that the HRE expert may have been more able to focus on the HRE task than were the intermediates. Furthermore, our results showed that the problem-solving processes of the expert and the intermediates differed slightly. We assume that the expert had a broader prior knowledge of Python programming. However, the large number of assigned open codes to both the intermediates and the expert in terms of Code Development indicated a general similarity of process.

Following the initial analysis of differences between the two groups in HRE Problem solving, two further questions derive, which are answered below: Was the expert able to solve the HRE task faster and more time-efficiently than the intermediates? Did the expert use different strategies than the intermediates?

4.3.5 Results RQ2

RQ2: Can differences in the time-efficiency of applied problem-solving strategies be identified?

In terms of HRE, efficiency is defined as a function of time as an analyst will eventually succeed if enough resources (e.g., devices, money) are given. We could also see this in the results of our study. All participants successfully solved the HRE task – but with solution times that ranged between 163 and 528 minutes (see Figure 7).

As described in the previous section answering RQ 1b, we found differences between the expert and intermediates mainly in the sub-categories, Inspection and Information Gathering, Reversing Strategy Decisions, and Reversing Problems. In order to answer RQ 2, we therefore focused on open codes that were assigned to these sub-categories. These open codes formed the basis for evaluating the efficiency of the participants' problem-solving strategies.

Describing problem-solving strategies based solely on the absolute number of open codes assigned was, in our view, not sufficient to derive conclusions about efficiency. Including the relative time (see Methods) that participants took to solve the HRE task allowed us to analyze participants' problem-solving processes in a fine-grained way.

Figure 7 presents the most dominant steps for each participant in the three above listed sub-categories. In addition, Table 8 shows how time-consuming each step was and at what point in each participant's timeline they were observed. Based on the tabulation of problem-solving processes, a case-by-case description of each participant's problem-solving strategies will follow in the next section. We structured the description based on the solution times and started with the description of the fastest participant.

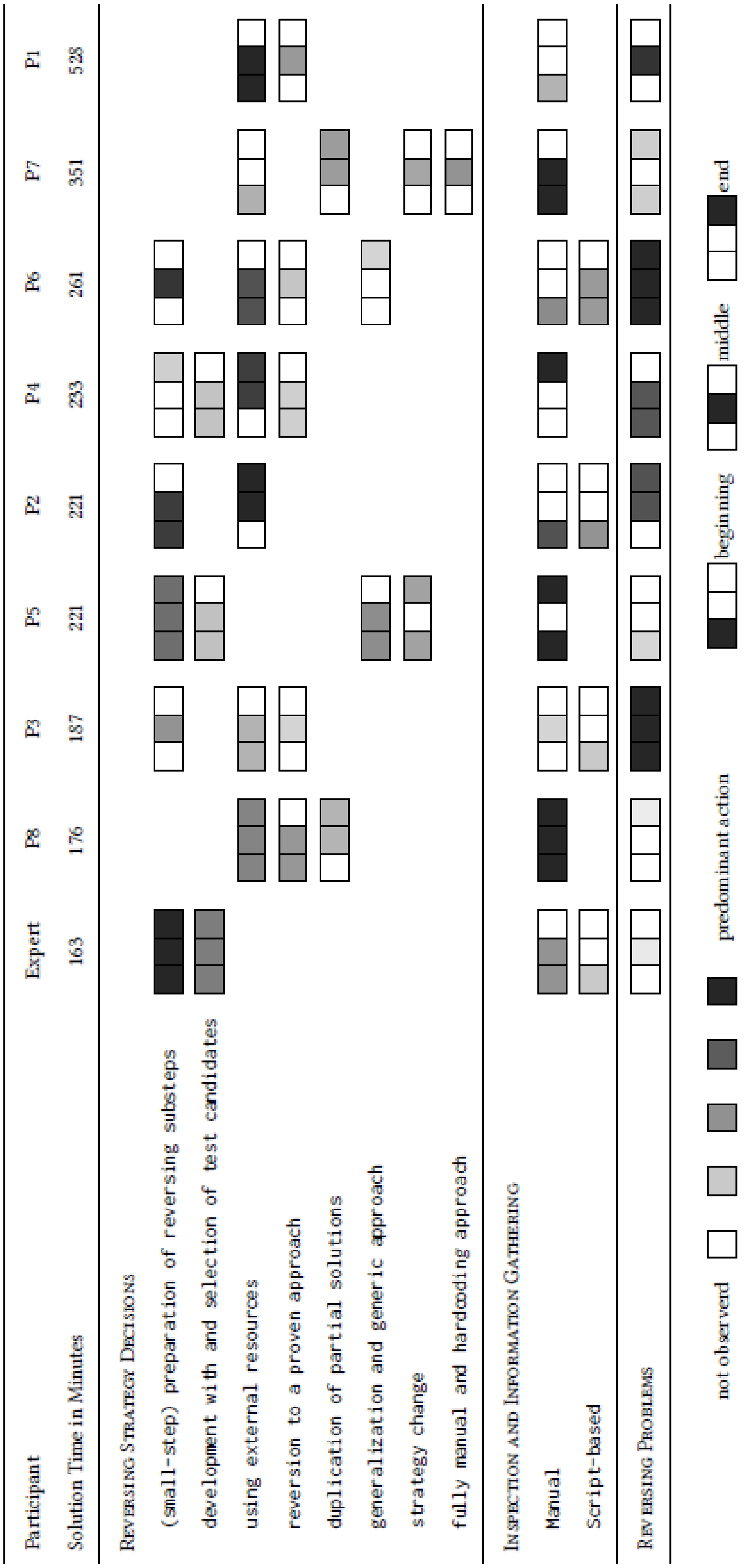


Figure 7. Strategy steps and solution times of the participants. Boxes on the bottom left: Darker values represent stronger focus on the respective action. Boxes on the bottom right symbolize whether the step occurred in the beginning, middle or end of the HRE problem-solving process.

Expert: Preparation of Reversing Sub-Steps, Development of Test Cases

The HRE expert solved the task in 163 minutes and was thus the fastest participant. In the sub-category Reversing Strategy Decisions, the open code *(Small-step) preparation of reversing sub-steps* emerged as the most dominant open code during the expert's HRE problem solving. We hypothesize that the expert divided the HRE task into smaller sub-tasks and prepared and processed these individually. This is also supported by the continuous occurrence of the open code *Development of test cases* that helped the HRE expert in continuously evaluating the solutions of subtasks. Furthermore, we found that the HRE expert applied both manual and script-based actions in the Inspection and Information Gathering sub-category. This showed that the expert was able to quickly transform the manual netlist analysis into scripts and thus, automated the information gathering process from the beginning. During the middle of the HRE problem solving, the expert faced a reversing problem as the expert lost track of the reversing approach that could be solved very quickly by manual analysis.

Participant 8: External Resources, Reversion to Proven Approach, Duplication of Partial Solutions

Participant 8 (P8) solved the HRE task in 176 minutes, what was very close the expert's solution time. Dominant in the approach of P8 were problem solving steps that could be described by the following open codes: *Using external resources, Reversion to a proven approach,* and the *Duplication of partial solutions* of the sub-category Reversing Strategy Decisions. Especially at the beginning and in the middle, we observed that P8 applied knowledge and skills that P8 had acquired from the previous HRE training tasks of the HRE training (*Reversion to proven approaches*: e.g., reversing methods to identify netlist components of interest). P8 continuously used external resources, such as the provided coding guide, pen and paper analyses, or online resources to solve the HRE task. Furthermore, we observed that P8 divided the HRE task into several similar sub-tasks, and applied the solution of the sub-tasks to solve other open sub-tasks (as reflected by the open code *Duplication of partial solutions*). In the context of problem-solving steps within the sub-category Inspection and Information Gathering, we observed that P8 performed only manual actions. While P8 conducted most of those manual information gathering processes during the beginning of HRE, we identified that P8 conducted further in-depth manual inspection of watermark candidates throughout the HRE task (reflected by the occurrence of the open code *In-depth manual inspection of watermark candidates*). In terms of problem-solving steps in the sub-category Reversing Problems, we found that P8 had to solve a very short-lasting phase of confusion (represented by the occurred open code *Lost track of the reversing approach*) that lasted 7 minutes, and was mainly caused by several simultaneous actions of code development.

Participant 3: External Resources, Preparation of Reversing Sub-Steps

Participant 3 solved the HRE task in 187 minutes and was very close to the expert's solution time. In terms of problem-solving steps in the sub-category Reversing Strategy Decisions, we observed that P3 often engaged in the preparation of reversing sub-steps (reflected by the open code *(Small-step) preparation of reversing sub-steps*). The preparation steps occurred during the first half of the HRE problem-solving process, and indicated that P2 divided the HRE task into smaller sub-tasks. In the beginning of P3' HRE problem solving, P3 used external resources (reflects by open code *Using external resources*), and script-based methods in terms of the sub-category Inspection and Information Gathering. By analyzing the netlist at the beginning, P3 laid the foundation of their problem-solving process and was not forced to gather more information as the process progressed. In addition to the stringent HRE problem-solving performance, we also noticed longer phases in that P3 faced reversing-specific problems and difficulties. The phase in that P3 was engaged in solving the reversing problem lasted 75 minutes, and included difficulties mainly from the cluster Lack of Understanding.

Participant 5: Development of Generic Approach, Development of Test Cases, Preparation of Reversing Sub-Steps

Participant 5 (P5) solved the HRE task in 221 minutes - about an hour longer than the HRE expert. In the beginning of the HRE problem solving, P5 used external resources (represented by occurred open code *External resources*), and script-based netlist analysis method from the sub-category Inspection and Information Gathering. Dominant in P5's actions observed in the sub-category Reversing Strategy Decisions was the development of a generic solution (reflected by the open code *Generic approach*). In general, a generic approach is characterized by its applicability to solve similar HRE tasks on other netlists. We detected the development of a generic approach in this extent only in P5' problem solving. Furthermore, we observed that P5 developed test cases (reflected by the open code *Development of test cases*) and selected test candidates (reflected by the open code *Selection of test candidates*) to solve the HRE task.

Additionally, we observed that P5 also engaged in the preparation of reversing sub-steps (reflected by the open code *(Small-step) preparation of reversing sub-steps*). In P5’s problem-solving process, we also detected a relatively short time (12 minutes) when P5 faced two reversing-specific problems (reflected by open code *Dead end*) and also changed their strategy for script-based analyses (reflected by open code *Strategy changes for script-based analyses*).

Participant 2: Preparation of Reversing Sub-Steps, External Resources

Participant 2 (P2) needed 221 minutes to solve the HRE task. Dominant in P2’ problem-solving process were the preparation of reversing sub-steps (reflected by the open code *(Small-step) preparation of reversing sub-steps*), and the application of external resources (reflected by the open code *External resources*). In the beginning of the HRE problem solving, we observed that P2 conducted a series of manual and script-based to explore the netlist and were assigned to the sub-category Inspection and Information Gathering. In the second half of P2’ problem solving, we identified that P2 had to solve reversing-specific problems (reflected by the cluster Lack of Understanding) that lasted for 35 minutes of the total solution time.

Participant 4: External Resources

Participant 4 (P4) was able to solve the HRE task in 233 minutes, and used external resources (reflected by the open code *External resources*) most frequently during the HRE problem-solving process. However, as the problem-solving process progressed, we identified that P4 also applied problem-solving steps such as the development of test cases (reflected by the open code *Development of test cases*), the reversion to a proven approach (reflected by the open code *Reversion to a proven approach*), or the preparation of sub-steps (reflected by the open code *(Small-step) preparation of reversing sub-steps*). We observed that P4 solely used manual netlist exploration and analysis steps (sub-category Inspection and Information Gathering). Furthermore, P4 had to solve several reversing-specific problems that lasted a total of 65 minutes. Our analysis showed that these reversing-specific problems occurred mainly during the beginning and the middle of the HRE problem-solving process and included actions from the cluster Failed Attempts and assigned open codes such as *Unsuccessful transfer of an already known approach to a current problem* or actions from the cluster Confusion with assigned open codes such as *Lost track of the reversing approach*.

Participant 6: External Resources and Preparation of Reversing Sub-Steps

Participant 6 (P6) solved the HRE task in 261 minutes. Dominant in P6' problem solving were actions from the sub-category Reversing Strategy Decisions with the assigned open codes *Using external resources*, and *(Small-step) preparation of reversing sub-steps*. In addition to these strategies, we observed the open codes *Reversion to a proven approach* and the development of a *Generic approach* towards the end of P6' problem solving. In the context of the sub-category Inspection and Information Gathering, our analysis showed that P6 applied several manual and script-based techniques at the beginning and during the middle of HRE. Furthermore, we observed that P6 faced and attempted to solve reversing-specific problems for 78 minutes. For example, P6 made several attempts to solve a Lack of Understanding.

Participant 7: External Resources, Fully Manual and Hardcoding Approach

Participant 7 (P7) was one of the slowest hardware reverse engineers and spent 351minutes to solve the HRE task. We observed that P7 developed the problem-solving strategy with external resources (reflected by open code *Using external resources*) in the beginning. As the problem solving progressed, we observed some unique approaches that we could not find in this form in any other analyst's problem solving. P7 changed the problem-solving strategy from script-based to manual (reflected by the open code *Strategy change from script-based to manual analysis)*. Furthermore, P7 decided to develop a fully manual solution (reflected by the open codes *Fully manual*, and *Hardcoding approach*) in the middle of the HRE problem solving. We assigned the open codes *Fully manual* and *Hardcoding approach* to problem-solving steps that included the encoding of fixed values into the solution that had been read out through manual netlist analysis. Additionally, in the context of Inspection and Information Gathering, we observed that P7 conducted solely manual netlist analysis (reflected by the open code *In-depth manual inspection of watermark candidates*). During the HRE problem solving, P7 encountered a relatively small number of reversing-specific problems, such as a Lack of Understanding} and Failed Attempts that could be solved by P7 in 35 minutes.

Participant 1: External Resources and Reversing Problem Shooting

Participant 1 (P1) was the slowest hardware reverse engineer and needed 528 minutes to solve the HRE task. Dominant in P1' problem solving was the inclusion of *External resources* and *Reversion to a proven approach.* During the second half of P1' problem solving, P1 failed to recognize that a correct solution of a sub-problem had been already reached. Due to this error, P1 inappropriately changed the reversing strategies. P1's approach in terms of Inspection and Information Gathering was based on manual analysis steps. P1 spent the longest overall time of all participants (167 minutes) to solve reversing-specific problems. These included assigned open codes such as *Correct solution is not recognized*, *Dead end*, *Introduction of a reversing-specific misconception*, and *Inappropriate change of the reversing strategy*.

4.3.6 Discussion RQ2

Our results did not indicate that a single optimal ("one and only" or "end all, be all") problem-solving strategy for most efficiently solving the HRE task existed. Rather, our analysis showed that several participants achieved efficient solutions through the application of different and individualized strategies. For example, we observed that the time-efficient problem-solving strategy of the HRE expert was based on test cases and small-step approaches that could not be observed to this extent in the problem solving of the intermediates. Furthermore, our results showed that the use of external resources was dominant in several problem-solving strategies of the intermediates. Using external resources, such as the provided coding guide, pen and paper analyses, or online resources may have supported the intermediates compensate for a lack of skills and knowledge in for example Python programming or reversing-specific concepts. Drawing upon external resources to solve the HRE task and obtain reassurance seemed to be an efficient approach for most intermediates.

In addition, we also found that the intermediates P5 and P7 applied unique HRE approaches, the former leading to a more and the latter to a less efficient solution. P5's problem solving was mainly characterized by the development of a generic approach that would have been useful to completing a similar reversing tasks on a different netlist. Nevertheless, it was relatively inefficient to solve the HRE task at hand. In contrast, P7's unique fully-manual approach caused an inefficient long solution time, but also included a relatively small number of reversing-specific problems. Based on this result, it is assumable that a step-by-step manual analyses and subsequent encoding of manually-identified netlist components was accurate, but also very time-consuming. In terms of Inspection and Information Gathering, we found that approaches that included both script-based and manual analysis steps, appeared to be more time-efficient than approaches that were solely based on manual analysis. Nevertheless, manual analysis also supported the hardware reverse engineers to solve reversing-specific problems.

Furthermore, our results indicated that more reversing-specific problems led to longer solution times. For instance, the two fastest participants (expert and P8) faced a single reversing-specific problem and were able to solve it within seven minutes. In contrast, P1 experienced a larger number of reversing-specific problems resulting in a longer time spent on task. However, our data revealed one exception: We observed a relatively large number of reversing-specific problems in P3' problem solving, who was one of the fastest participants. Although P3 faced longer periods of difficulties related to the cluster of Lack of understanding, P3 was able to achieve important reversing milestones and did not lose the golden threat.

Based on our previous assumptions and the results of RQ1, it would have been plausible to assume that the HRE expert would have been the most efficient participant. In contrast to these expectations, our in-depth analysis of applied problem-solving strategies and their time-efficiency showed that beside the HRE expert, two intermediates also achieved very time-efficient solutions. We discuss this finding in greater detail in the General Discussion (see Chapter 5).

4.3.7 Results RQ3

RQ3: Do cognitive abilities play a role in HRE problem solving?

Due to the small size of our sample, statistical analysis of the influence of cognitive factors on HRE performance was not feasible. However, we found one interesting descriptive result on the role of working memory in the time-efficiency of HRE problem solving. The descriptive data suggested a potential negative correlation between working-memory scores and the overall time on task in the HRE task (see Figure 8). The result leaded to the assumption that participants with higher working-memory scores tended to solve the HRE task quicker than participants with lower working-memory scores. Participant 8 (P8) with a high score in working memory (126) achieved a short solution time of 176 minutes. Contrary, P1 with a lower score in working memory (108) had a longer solution time with 528 minutes. Please note, that in the paper

(Becker et al. 2020) the solution time of P1 was calculated with 398 minutes. Based on our detailed analysis with the iterative open coding (Strauss & Corbin, 1998), we detected that P1 developed an algorithm during an idle-phase. This idle phase was not involved in the calculation of P1's solution time within our publication (Becker et al., 2020), since we assumed at this point that no HRE-specific actions happened in the corresponding idle phase. P6 seemed to be an outlier that is discussed in the following section. Table 7 summarizes the descriptive data of cognitive factors and solution time in the HRE task.

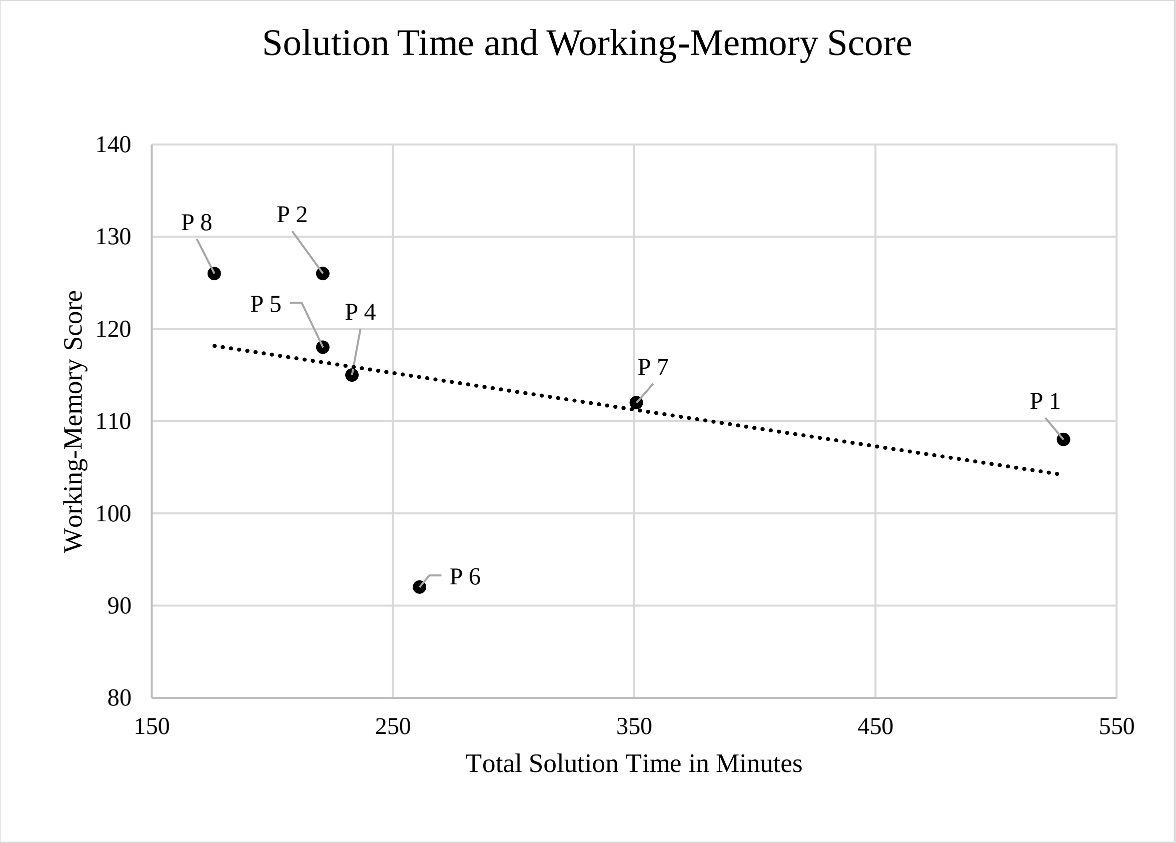


Figure 8. Scatter plot of solution time (x-axis) and working-memory scores (y-axis). Note: P3 and expert did not participate in the cognitive tests.

Table 7. Time spent on the HRE task (minutes) and scores of the cognitive factors working memory (WM), processing speed (PS), and perceptual reasoning (PR).

Variable	Participants								
	P1	P2	P3	P4	P5	P6	P7	P8	Expert
Time	528	221	187	233	221	261	351	176	163
WM	108	126	-	115	118	92	112	126	-
PS	106	146	-	146	119	109	119	117	-
PR	104	129	-	100	115	100	104	106	-

Note. P3 and the HRE expert did not participate in the WAIS-IV

4.3.8 Discussion RQ3

The descriptive results of RQ3 suggested a potential negative correlation between working-memory scores and solution time in solving the HRE task. Due to small sample sizes we conducted a descriptive analysis and hypothesized that participants with higher working-memory scores tended to solve the HRE task faster than participants with lower working-memory scores. In the following, I describe our assumptions in more detail.

The term working memory is defined as a cognitive system that explains how information are stored and processed simultaneously (Baddeley, 1992). In more detail, working memory is responsible for the storage of sensory input (e.g., visual information) in immediate awareness and the manipulation of that sensory input to solve complex task such as problem solving (Baddeley, 1992). Baddeley and Hitch (1974) proposed the model of working memory that originally consisted of three main parts: visuospatial sketchpad; phonological loop, and the central executive (Baddeley, 2002). The visuospatial sketchpad is responsible to temporarily store and process visuospatial information and is essential for solving visual problems (e.g., visual analysis of complex netlist during HRE) (e.g., Baddeley 2002). The phonological loop is the counterpart to the visuospatial sketchpad and is important for the temporary storage and processing of acoustic information (e.g., Baddeley 2002). An individual’s attention it directed to relevant information and to coordinate cognitive processes due to the central executive of the working memory (e.g., Baddeley 2002). Prior research showed that individual differences in working-memory tests were commonly determined by the individuals’ executive processes

(Daneman & Carpenter, 1980; Baddeley, 2003). In 2000, a fourth component the episodic buffer was added to the working memory model (Baddeley, 2000). The episodic buffer is hypothesized to build an interface between the sub-systems (i.e., visual sketchpad) and the long-term-memory (LTM) (Baddeley, 2000). According to Baddeley (2012), it can be described as a buffer store through that working memory is connected to perception and LTM. It is assumed to have limited capacity (Baddeley, 2012) with four chunks (Cowan, 2005).

According to Diamond (2013), working memory is central for sense-making processes and to solve complex cognitive tasks. The functionalities of the working memory extend to for example, i) inclusion of new information in currently existing thought and action processes; ii) development of alternatives; iii) mental connection of information to establish general principles or to identify connections between individual aspects (Diamond, 2013). Furthermore, working memory is supposed to support inhibitory control in goal-directed actions (Diamond, 2013), and to cognitive abilities of inhibiting specific information (Hasher & Zacks, 1988; Zacks & Hasher, 2006). By keeping the goal in mind and by inhibiting irrelevant information, an individual increases the efficiency of their actions (Diamond, 2013).

We hypothesize that several functionalities of the working memory may have contributed to achieve time-efficient solutions in HRE and describe our assumptions in the following. First, the time-efficient solution of participants with higher scores in working memory may be based on the inhibitory functionalities of the working memory (e.g., Diamond, 2013). Especially during the visual netlist exploration and analysis phases, hardware reverse engineers had to handle a wealth of visual information and have to distinguish between relevant and irrelevant visual stimuli. We suggest that participants with stronger working memories were more able to inhibit irrelevant stimuli (e.g., irrelevant netlist components) and in storing and manipulating visual information within the visuospatial sketchpad than participants with lower working-memory scores. The efficient cognitive inhibition of those participants with higher working-memory scores enabled them to delete irrelevant information from the limited working memory's capacity, and to still focus on the reversing goal (Hasher & Zacks 1988, Zacks & Hasher 2006; Diamond, 2013).

Second, we assume that the working memory functionality of efficiently holding and manipulating information (e.g., Baddeley, 2002) may have played a role in solving the HRE task efficiently. The analysis of specific netlist components and interconnections is very complex due to a variety of different information. It is plausible to hypothesize that a stronger working memory enabled participant to identify connections between single components and to develop goal-directed action plans without losing the golden threat (i.e., the reversing goal of identifying and extracting watermark candidates).

Third, we assume that the functionality of working memory of activating and retrieving information from LTM (e.g., Baddeley, 2012) to solve the present task, may have been supportive during HRE problem solving. Although both the fastest and the slowest participants acquired the same amount of HRE knowledge and HRE skills (e.g., methods to read out data from the identified candidates) during the training phase of the HRE course, P8 was quicker in solving the HRE problem. As previously outlined, working memory activates and retrieves already stored information from LTM (e.g., Baddeley, 2012). Thus, we assume that the stronger working memory supported P8 to efficiently activate and retrieve stored information from the LTM in order to analyze and work with inputs P8 stored in immediate awareness in the working memory.

Finally, the results revealed an outlier (P6) with the lowest working-memory score. It may be plausible to assume that measurements of working memory abilities of P6 could have been influenced by uncontrolled variables. Prior research has revealed that several variables can impar performance in working memory tasks, for example, chronic psychological stress, chronic psychological stress (Mizoguchi et al., 2000), acute psychological stress (Qin, Hermans, van Marle, Luo, & Fernández, 2009), negative emotions like anxiety (Moran, 2016), or neural disorders (Willcutt, Doyle, Nigg, Faraone, & Pennington, 2005). This could mean that the measurement of working memory of P6 may have been negatively influenced, resulting in a worse value than actually exists.

5 General Discussion

Disclaimer: *Parts of this chapter were adapted from the submitted paper "The Anatomy of a Hardware Reverse Engineering Attack: Insights into Cognitive Processes during Problem Solving" that is currently under review at the journal ACM TOCHI in 2020 (Wiesen, Becker, Walendy, Paar, & Rummel, 2020). This paper was written together with my co-authors Steffen Becker, René Walendy, Christof Paar, and Nikol Rummel.*

General Discussion – HRE as a Function of both Expertise and Cognitive Abilities?

The results of my dissertation point to the role of expertise and cognitive abilities for time-efficient problem solving during HRE. In this chapter, I discuss the two main contributions of my thesis based on our results and the literature. First, findings of this doctoral thesis contribute to the ongoing debate in psychology about the role of expertise and cognitive abilities in problem-solving performance. In this context, I make a theoretical contribution by concluding that HRE problems may involve aspects of both simple and complex problems. Second, my discussion follows the considerations of Lee and Johnson-Laird (2013), who outlined that reverse engineering of Boolean systems may be a specific type of human problem solving.

Our study showed that the HRE expert was able to achieve the quickest and most time-efficient solution in the HRE task by applying a unique set of problem-solving strategies not used in this form by any other analyst. This aligned with findings from the domains of solving physics problems (Larkin, McDermott, Simon, & Simon, 1980) and medicine (Patel & Kaufman, 1995), showing that experts were more successful in selecting and using an appropriate strategy than non-experts (Chi, 2006). Lemaire and Siegler (1995) found that experts often used strategies that have proven to be effective in previous problem-solving situations. Ericsson and colleagues (1993) suggested that the superior performance of experts was based on experts' wealth of domain-specific knowledge and prior experience in solving domain-specific problems that they had accumulated throughout several years of deliberate practice. Additionally, experts' well-structured and domain-specific knowledge significantly influenced the perception and categorization of problems, which in turn, led to efficient problem-solving performance (Nokes et al., 2010; Chi, Feltovich, & Glaser, 1981a). The categorization and representation of problems supported experts in selecting the most suitable problem-solving strategies and, thereby, they generated time-efficient solutions (Chi et al., 1981; De Groot, 1978; Patel & Groen, 1986). Against this background, it is reasonable to assume that the HRE expert in the present study selected problem-solving strategies based on a wealth of HRE-specific knowledge. In turn, this domain-specific knowledge may have supported the HRE expert in their highly efficient problem categorization and representation, that enabled the expert to achieve a time-efficient solution. Ericsson and Kintsch (1995) explained the efficient problem solving of experts by the hypothesis that experts developed an effective long-term working memory that enabled experts to efficiently activate prior knowledge (e.g., in form of stored procedures or chunks) from the LTM and thus, may supported them in circumventing limited capacities of the working memory. Thus, the HRE expert's problem solving may have been based on an efficient activation and retrieval of prior knowledge from the LTM.

In addition, our results indicated that besides the HRE expert, two intermediates (P8, P3) achieved time-efficient solutions even though they had less domain-specific knowledge and less problem-solving experience than the HRE expert. Furthermore, we found that both intermediates (P8, P3) had above-average scores in the intelligence-subfactor working memory. We assumed that the highly efficient solutions of these two intermediates were based on their cognitive abilities (e.g., working memory) that supported them to compensate for gaps in domain-specific knowledge. Our assumption aligned with the expert-performance framework that explained how novices and intermediates cope with gaps in domain-specific prior knowledge and problem-solving experience (Ericsson et al., 1993). Ericsson (2014) hypothesized that a correlation between domain-specific performance and performance on tests of general cognitive ability may exist for non-experts, as shown by prior research of Ackermann (1987), or Schmidt and Hunter (2004). Consequently, HRE problem solving of novices and intermediates may be correlated with general cognitive ability and may explain that intermediates P8 and P3 achieved time-efficient solutions in HRE due to their above-average levels of working memory. In contrast to both intermediates, the HRE expert may have acquired specific knowledge structures throughout years of deliberate practice that may have mediated their superior performance in the HRE task. As the expert did not participate in the intelligence test WAIS-IV (Wechsler, 2008), we cannot draw any conclusion about the correlation between the HRE expert's level of cognitive ability and their problem-solving performance.

In this context, we assumed that both intermediates may have been supported by their working memory in solving the HRE task. Our descriptive data showed that intermediates who achieved higher working-memory scores tended to solve the HRE task faster than intermediates with lower working-memory scores. Against this result, we suggested that specific sub-systems (such as the working memory; e.g., Baddeley & Hitch, 1974) of the intermediates' cognitive information processing system supported the high-performing intermediates in the current study to temporarily keep information in mind and to work on it. The inclusion of novel information in currently existing thoughts or action processes to solve complex cognitive tasks, is one of the central functionalities of the working memory (e.g., Diamond, 2013). Including novel information in existing problem-solving plans, may have been supportive in the HRE problem solving of both intermediates P3 and P8. We assumed that their above-average working memory may have helped them to efficiently integrate novel information about specific netlist components and their interconnections in their existing problem-solving plans. Furthermore, our analysis of the fast intermediates' strategies revealed that the intermediates used procedures that they had acquired in previous HRE training tasks. As shown by prior research working memory supported the activation and retrieval of stored information from LTM to solve the task at hand (e.g., Baddeley, 2012). Thus, we assumed that the working memory supported P8 and P3 to efficiently activate and retrieve stored information from the LTM to analyze and work with inputs that the participants stored in immediate awareness in the working memory (see detailed discussion in Section 4.3.8).

The influence of prior knowledge and cognitive abilities on problem-solving performance lies also at the heart of taxonomies that define human problem solving. Prior psychological research established a broad range of taxonomies to define problems, and typically distinguished between simple and complex problems (e.g., Dörner & Funke, 2017). Solving simple and solving complex problems relies on different cognitive processes (Schraw, Dunkle, & Bendixen, 1995; Dörner & Funke, 2017). Complex problems are usually described as knowledge-rich systems that activate large semantic networks of prior knowledge and potentially successful problem-solving strategies (Dörner & Funke, 2017). Simple problems are typically described as knowledge-free systems because solving simple problems, opposed to solving complex problems, is less based on domain-specific knowledge (Chi, Glaser, & Rees, 1981b) and more on cognitive abilities (e.g., Sternberg, 1982). It remained an open question to which problem type HRE belongs (see Background).

We hypothesize, that HRE tasks seem to involve aspects of both simple and complex problems. In the beginning of an HRE problem-solving process, the reverse engineer obtains a gate-level netlist that can be defined as the initial state of the HRE problem. By selecting suitable operators (e.g., information gathering based on manual netlist analysis) the analyst tries to achieve a desired goal state (e.g., removal of watermark from a protected circuit) or sub-goals derived from the main goal. An unambiguous initial state and the goal state as well as a clear set of operators that allow to move in the problem space are common characteristics of simple problems (Dörner & Funke, 2017). Since HRE tasks neither change independently nor because of inputs from the problem solver, they can be described as time stable, which is a common characteristic of simple problems (Funke, 2003). In contrast to simple problems that can be solved without domain-specific knowledge, a realistic HRE task may be impossible to solve for a person with insufficient domain-specific knowledge and skills (e.g., on hardware circuits, chip design, etc.). Another characteristic of HRE tasks is the amount of information in the netlist that must be analyzed and processed. Within the scope of netlist analysis, a hardware reverse engineer analyzes from thousands up to millions of components and their interconnections. The complexity of such an HRE task increases with netlist size (i.e., the number of components) that requires reduction and abstraction. Such characteristics of complexity and connectivity found in HRE tasks are commonly assigned to complex problems (e.g., Funke et al., 2018). Additionally, especially at the beginning of the HRE process, not all the necessary information is apparent to the analyst from the start. Thus, an analyst needs to perform a series of operations that lead to obtaining the needed information. Consequently, the hardware reverse engineer tries to remove a lack of transparency at the beginning of HRE, which is commonly considered as a characteristic of CPS (e.g., Funke, 2012). While these three aspects (i.e., complexity, connectivity, and non-transparency) of complex problems can be found in HRE problem solving, the additional two characteristics of complex problems – dynamics and polytelic situations – are absent from HRE problems. In contrast to complex problems that usually change dynamically, HRE problems are stable. Furthermore, a reverse engineer analyzes the netlist components to achieve (sub-)goals and a goal state that are clearly defined, non-competitive and not mutually exclusive. The embedding of HRE problem solving in existing problem-solving taxonomies leads to assume that HRE tasks may combine aspects of both simple and complex problems. Although HRE and simple problems have several core aspects in common (e.g., unambiguous initial state; the goal state), the huge amount of information pertaining to non-transparent, highly interconnected components that must be processed and analyzed, creates a challenge that places HRE squarely outside of the realm of simple problems as typically defined and constructed. These characteristics add another layer to HRE problems and thus places them between simple problems and complex problems.

Our results relate also to the findings on problem solving in reverse engineering of Boolean systems by Lee and Johnson-Laird (2013), who suggested that reverse engineering of Boolean systems may be a specific type of human problem solving. As previously outlined (see Background), it was unclear to what extent the results of Lee and Johnson-Laird (2013) concerning applied problem-solving strategies and difficulties in reverse engineering of Boolean systems were applicable to the domain of HRE problem solving. Lee and Johnson-Laird (2013) showed that participants applied one of two main problem-solving strategies in drawing circuits that would control an electrical light or a water flow system: Either the participants focused on single outputs at a time, or on a single component at a time. In contrast to Lee and Johnson-Laird (2013) we did not identify those two main categories during HRE problem solving, and found that none of the applied problem-solving strategies could be described as the single “best” or as the main strategy for solving the realistic HRE task. While there was some overlap in applied sub-strategies (e.g., using external resources), it is important to note that most participants used a unique strategy. Furthermore, our results showed that some participants (e.g., generic approach of P5; manual approach P7) developed problem-solving strategies that were almost incomparable to other approaches. Accordingly, it is unclear whether we could and should divide HRE strategies into strategy categories. It might be the case that hardware reverse engineers solve HRE tasks in their own way or set different emphases that lead to only a few overlapping strategy characteristics.

Furthermore, Lee and Johnson-Laird (2013) suggested that difficulties in reverse engineering of Boolean systems depended on three factors: a) number of components, b) number of components influencing an output, and c) dependencies of components that influence an output. It is plausible to assume that those three factors may have also influenced the level of difficulty of solving the HRE task. Especially the dependence of single netlist components to other netlist components may have influenced the perceived difficulty of the HRE task. Accordingly, the more components there are and the more they are interconnected, the more difficult and non-transparent it could be for a hardware reverse engineer to analyze a netlist. Based on our in-depth analysis of realistic HRE problem solving, we extended the understanding of reversing-specific difficulties that occurred during HRE problem solving. For example, we found difficulties, such as dead-ends that were caused by misleading strategies that may have led to longer solution times in some cases (e.g., P1; P6).

Taken together, our results suggested that both the HRE expert and two intermediates with above-average scores in working memory achieved superior performance. This result let me assume that HRE problem solving may be a function of both cognitive abilities and expertise. This was in line with the assumptions of the expert-performance approach (Ericsson et al., 1993), proposing that domain-specific problem solving was correlated with cognitive abilities in non-experts and with prior domain-specific knowledge in experts. In this context, I discussed that the fast and efficient solution of the HRE expert was based on well-structured domain-specific knowledge. Furthermore, we proposed that the superior HRE performance of both intermediates was based on their above-average working memory that may have supported them in solving the cognitively complex HRE task at hand. In this context, I made a theoretical contribution by concluding that HRE problems may involve aspects of both simple and complex problems. Furthermore, our results were also aligned with a debate initiated by Lee and Johnson-Laird (2013) who suggested that reverse engineering of simplified Boolean systems may be a specific type of human problem solving, which remained so far poorly understood. I discussed to what extent the results by Lee and Johnson-Laird (2013) were applicable to the domain of HRE problem solving. My main contributions provide stimuli for future research (see Section 7) and for further considerations for cognitive obfuscation impeding HRE (see Chapter 6).

6 Initial Ideas for Cognitive Obfuscation

In general, countermeasures such as obfuscation techniques never completely protect against HRE (Barak et al., 2001). The underlying principle of obfuscation is to transform the ICs of a microchip so that high-level information is obstructed and functionalities of the ICs do not change (Wiesen et al., 2019a). As HRE includes both technical and cognitive processes, we suggested that countermeasures that aim to impede HRE should also focus to impede the cognitive processes in HRE – suggested by us as cognitive obfuscation (Wiesen et al., 2019a; Becker et al., 2020).

In this section, I develop initial ideas for future studies to pave the way for the development of cognitive obfuscation techniques. Based on our results, I will focus on the cognitive aspects of HRE and hope to provide impulses for future studies on the development of cognitive obfuscation prototypes. In particular, based on one of my main contributions (i.e., HRE may be a function of both cognitive abilities and expertise), I develop initial ideas for cognitive obfuscation that may impede HRE problem-solving processes that are based on cognitive abilities and on expertise.

6.1 Overloading the Capacity of the Working Memory

The results of this doctoral thesis suggested that the working memory of hardware reverse engineers may play an important role in efficiently solving HRE tasks. Descriptive data suggested that participants with higher working-memory scores solved the HRE task with shorter solution times than participants with lower working-memory scores. In general, working memory is responsible for storing, retrieving and processing information to enable goal-directed behavior, for example, in cognitively challenging tasks, such as problem solving (e.g., Baddeley, 2012). Thus, it is plausible to assume that working memory supported hardware reverse engineers in processing the wealth of information about netlist components and their interconnections during HRE. This leads to ask, whether cognitive obfuscation may be able to overload the capacity of analysts' working memories.

Prior research has shown, that the support of working memory in solving cognitively demanding tasks is restricted by the generally limited capacity of the working memory. As the number of information determines how many chunks (i.e., group of meaningful information cues) can be stored and processed in the episodic buffer of the working memory, it is difficult to define the capacity of working memory by a concrete number of chunks. Nevertheless, prior research suggested that the capacity of the episodic buffer of the working memory may be limited to four chunks (Cowan, 2005; Baddeley, 2012). In addition, Miller (1956) suggested that the capacity of individuals to process information is limited to seven (plus or minus two) objects. Against this background, the limited capacity of working memory may be a first starting point for cognitive obfuscation. Specifically, if cognitive obfuscation can achieve that the working memory of hardware reverse engineers is overloaded, it may be possible that HRE performance degrades.

In this context, prior research showed that participants' performance in a visuospatial task was impeded by a concurrent visuospatial task or by a task that asked to generate random digits (Baddeley, Emslie, Kolodny, & Duncan, 1998; Baddeley, 2002). Those concurrent tasks heavily overloaded the capacity of the central executive of the working memory (Baddeley et al., 1998; Baddeley, 2002). In terms of HRE, I suggest that cognitive obfuscation may act as a concurrent task that may lead to interrupt the solving of the main HRE task. Thus, the analysts may be forced to divide their attention to solve the main and the concurrent tasks that may lead to an overload of the working memory capacity. I will present my idea in the following.

An interruption is defined as a secondary activity or task that draws an individual's attention away from the primary task (Li, Magrabi, & Coiera, 2012). Concurrent interrupting tasks are typically associated with costs in the form of additional time spent on task (e.g., time to return to the main task after completing the secondary task; Trafton, Altmann, Brock, & Mintz, 2003; Li et al., 2012) and reduced solution accuracy (e.g., probability of errors increases). The memory of goals framework describes how interruptions influence task performance and how performance is impeded by goal decay (Altmann & Trafton, 2002). More precisely, the goals of the primary tasks immediately start to suffer from activation decay when a concurrent interrupting task occurred (Monk, Trafton, & Boehm-Davis, 2008; Altmann & Trafton, 2002). In addition, prior research suggests factors that may aggravate the disruptive effects of interruptions (Monk et al., 2008): i) high similarity (e.g., Cellier & Eyrolle, 1992; Edwards & Gronlund, 1998; Oulasvirta & Saariluoma, 2004) and relatedness (Cutrell, Czerwinski, & Horvitz, 2001; Zijlstra, Roe, Leonora, & Krediet, 1999) of the main and the interruption tasks; and ii) complexity (i.e., cognitive demand) of the interruption (e.g., Cades, Boehm-Davis, Trafton, & Monk, 2007; Gillie & Broadbent, 1989).

In their meta-review, Li and colleagues (2012) summarized that an interruption with strong similarities to the main task is more likely to interrupt the completion of the main task than an interruption with lower similarity. For instance, Li and colleagues (2012) drew on an example from everyday medical work. They suggested that if a nurse who is filling out a medical record for patient A is interrupted to do the same documentation for patient B, it could be that patient A's documentation suffers. In turn, they suggested that if the nurse is interrupted with a non-similar task (e.g., inquiry about health insurance), this may have fewer negative effects on the main task. Furthermore, interruptions with higher similarities to the main task may be more likely to activate the same cognitive mechanism that are needed to solve the main task (Li et al., 2012). Consequently, an interrupting task with strong similarities to the main task may demand similar cognitive structures and may increase the goal decay and the time on task to solve the primary task.

Initial idea for cognitive obfuscation. In summary, our results suggest that working memory may play an important role in HRE problem solving. Thus, it may be a valuable approach to develop cognitive obfuscation that overwhelms specific cognitive structures and causes disruptive effects on HRE performance. Cognitive obfuscation that aims to overload the capacity of the working memory could include concurrent tasks that are very complex and characterized by a high similarity to the primary HRE task. Complexity and similarity have been shown to disrupt the problem solver and lead to decay of primary goals (e.g., Monk et al., 2008). I hypothesize that cognitive obfuscation tasks that include a high complexity and task structures that place similar cognitive demands as the primary tasks may cause an overload of the working memory. Ideally, obfuscation tasks cause long periods of time spent on the obfuscation task before an analyst can return to the primary task, that is, reverse engineering the chip. In turn, this may disrupt the problem-solving process for the primary HRE task, cause additional time on task, and potentially increase the likelihood of errors in the primary HRE task. Ideas for future research on this obfuscation task that aims to overload the capacity of the working memory are presented in the following chapter 7.

Nevertheless, prior research has also investigated how individuals can mitigate the disruptive effect of interrupting concurrent tasks. In this context, practice and prior experience were shown to counter disruption by interruptions (Li et al., 2012). In the following, I explain why it may not be sufficient to solely overwhelm the working memory and why it seems to be necessary to include other aspects in the development of cognitive obfuscation methods. Although, the capacity of the working memory may be not expandable by practice, memory training and knowledge structures (e.g. chunks) may expand the capacity to transfer and retrieve information from the long-term working memory (Ericsson & Kintsch, 1995). Especially experts with well-structured knowledge and large chunks may circumvent the limited capacities of working memory by retrieving and including relevant domain-specific knowledge and procedures with minimal cognitive effort (Alexander, 2003; Chi, 2006). Furthermore, experts also face lower cognitive effort due to automated problem-solving procedures (Schneider, 1985; Chi, 2006).

6.2 Developing Misleading HRE Challenges

As suggested by our results, cognitive abilities and expertise may support analysts in conducting HRE. The central question is how to impede hardware reverse engineers (e.g., HRE experts or more experienced analysts) who also rely on their well-structured prior knowledge. I suggest that two approaches could be pursued in this context: experts' mental fixations on applying standard strategies and experts' missing flexibility to develop non-routine solutions.

Besides several findings on how expertise supported problem solving (e.g., Larkin, McDermott, Simon, & Simon, 1980; Patel & Kaufman, 1995), prior research on expertise has also identified drawbacks that stem from increasing expertise (e.g., summary in Chi, 2006). Especially i) mental fixations through domain-specific knowledge that impede the development of creative and novel solutions and ii) experts' inflexibility to changing problem-solving routines and inability to adapt their strategies accordingly may be valuable to consider in terms of cognitive obfuscation.

One challenge that can impede the performance of high-knowledge participants are mental fixations on specific problem-solving procedures that do not necessarily provide the best solution for the task at hand. For example, Wiley (1998) showed in three experiments that individuals with higher levels of domain-specific knowledge were worse in solving a creative task than individuals with lower levels of domain-specific knowledge. Those creative tasks were characterized by misleading items in the domain of baseball. Therefore, Wiley (1998) adapted the Remote Associates Test (RAT) (Mednick, 1962) and asked participants to logically complete a series of three baseball words by a fourth. The adapted RAT-test included misleading tasks, in which the solution was not a baseball-related word (Wiley, 1998). The three experiments showed a clear effect that participants with higher levels in domain-specific baseball knowledge were less successful in solving the problems correctly as their prior knowledge caused a mental fixation on an inappropriate solution. Wiley (1998) argued that the high-knowledge participants did not conduct a broad search in the search space of the problem that may have influenced the fact that experts did not choose an appropriate solution. Since the low-knowledge participants achieved better solutions, Wiley (1998) suggested that participants with lower domain-specific knowledge were able to solve the misleading tasks more flexible than the high-knowledge subjects.

Furthermore, experts are outperformed by non-experts when a task included new contexts and demanded to develop and apply non-routine solutions. Prior research showed that experts were less flexible than novices in solving a domain-specific task for that new information or new contexts had to be considered (Chi, 2006). For example, Marchant, Robinson, Anderson and Schadewald (1991) showed that accountants with higher levels of domain-specific knowledge were less successful in applying new information (about a tax law) that impeded the application of standard deductions. In addition, Frensch and Sternberg (1989) presented that expert bridge players were less able to apply a new version of the game than novice players. Wiley (1998) concluded that these studies showed that highly proceduralized domain-specific knowledge may be responsible that experts were less flexible to new contexts and information.

Initial idea for cognitive obfuscation. Prior research has shown that domain-specific knowledge can enhance problem-solving processes on the one hand (e.g., Patel & Kaufman, 1995), but can also inhibit effective of problem-solving processes on the other hand (e.g., Wiley, 1998). There is no doubt that domain-specific knowledge is essential to solving problems. Nevertheless, there exists some empirical evidence that experts also tend to fail due to i) domain-specific knowledge that acts as a mental fixation impeding creative and novel solutions (Wiley, 1998) or ii) an inflexibility to adapt their strategies accordingly to new information or contexts (e.g., Marchant et al., 1991). Against this background, I suggest that cognitive obfuscation that aims at posing challenges for HRE experts should include tasks and problem settings that are unusual for the HRE domain. These non-routine obfuscation tasks may include misleading aspects that may force HRE experts to develop non-routine problem-solving strategies. I hypothesize that HRE experts may be less flexible in adapting their HRE problem-solving strategies in an appropriate way and may be more focused on applying their standard strategies. This may cause less efficient HRE problem-solving performance of HRE experts. Ideas for future research on this obfuscation task are briefly presented in the chapter 7.

In summary, I suggest that cognitive obfuscation tasks should include cognitively challenging tasks that impede both problem-solving processes determined by expertise and cognitive abilities. In particular, I suggest that analysts whose superior HRE performance is based on their highly-efficient working memory, could be disturbed by complex interrupting tasks with high similarities to the primary HRE task that may lead to inefficient problem solving. These concurrent and interrupting obfuscation tasks could activate similar cognitive structures that were already needed to solve the primary HRE task and could overload the capacity of the analysts' working memory. As shown by prior research with a visuospatial task (Baddeley et al., 1998; Baddeley, 2002), those concurrent obfuscation tasks could heavily overload the capacity of the central executive of the working that may lead to an inefficient HRE problem-solving performance.

Nevertheless, prior research has shown that more experienced individuals may mitigate the disruptive effects of concurrent interrupting tasks (e.g., Li et al., 2012) by training and experience. Thus, overloading the capacity of working memory may not be enough to impede HRE problem solving by analysts with a profound level of domain-specific knowledge and prior problem-solving experience. In a second step, I developed an initial idea how high-knowledge individual may fail in achieving an efficient HRE solution. Prior research has shown that when a problem asked for non-standard solution paths that lay outside their standard problem-solving procedures, experts were less efficient in adapting and changing their problem-solving strategies (e.g., overview in Chi, 2006). Thus, it may be assumable, that cognitive obfuscation tasks that include sub-tasks that are unusual or new to the HRE domain and cannot be solved by standard strategies, might force analysts to develop new strategies that may raise the time spent on task and lead to inefficient HRE problem solving.

7 Limitations and Future Studies

While this thesis provides valuable insights into HRE problem-solving processes and relevant cognitive factors such as working memory or expertise, it is not without certain limitations. Based on these limitations, I will derive ideas for future research below.

The main study was limited by a small sample size of eight intermediates and one HRE expert. The qualitative nature of this work and the application of an iterative open coding scheme based on Grounded Theory (Strauss & Corbin, 1998) enabled the analysis of problem-solving processes and strategies exhibited by the participants in greater detail. Nevertheless, HRE problem-solving processes and correlations with cognitive factors could only be described in a qualitative manner, and we could not draw any conclusions about whether working memory or level of expertise had a significant influence on problem-solving performance in HRE. Thus, I propose that future studies should quantify HRE problem-solving processes and the influences of cognitive factors with larger sample sizes. Furthermore, a promising future investigation would be to analyze the frequencies of reversing-specific problems and difficulties during several HRE tasks, and whether these could be avoided by hardware reverse engineers as their experience grows.

Furthermore, we circumvented the methodological problem that HRE experts were unavailable for research, by the development of an HRE course that promoted HRE skill acquisition in students with relevant backgrounds. Although our methodological approach enabled us to investigate HRE problem solving, it also comes with some limitations. Although, we successfully conducted our main study, we were only able to recruit a small number of participants. In addition, all students came from the same university and also completed the same HRE training. Although we aimed to avoid biases as much as possible (e.g., by not making concrete solution scenarios or solution strategies for the HRE task available to students), it is important that future studies recruit a more heterogeneous sample for analyzing human problem solving in HRE. One idea, we are currently pursuing, is the development of an HRE simulation. This simulates realistic sub-processes of HRE in a knowledge-free environment and thus may enable the recruitment of a broader and larger sample (i.e., different scientific backgrounds; different levels of cognitive abilities; different levels of HRE expertise). Our HRE simulation will also enable us to quantify influences of cognitive abilities and expertise on HRE problem-solving performance.

Furthermore, the HRE problem-solving model and qualitative analysis were based on a single realistic HRE task. While, we believe that our model captures the most significant HRE processes, it is still necessary to further explore reverse engineers' problem-solving processes in further HRE tasks, and to examine whether our model is valid for other HRE tasks (e.g., identification of an FSM in a netlist).

In addition, I developed initial ideas for cognitive obfuscation, which in turn should be understood more as a first impulse and less as concrete suggestions. One idea describes how it would be theoretically possible to confuse experts or persons with a high level of domain-specific prior knowledge and to negatively influence their HRE problem-solving performance. In this context, it would be valuable if future studies could recruit more HRE experts in order to analyze their problem-solving processes and to investigate, which problem-solving strategies experts typically apply. Furthermore, it would be valuable to explore, if HRE experts would be able to adjust their strategies to uncommon or changing obfuscation tasks. A second idea was to develop obfuscation tasks that may overload the capacity of the working memory of analysts. Prior research suggested that concurrent tasks with high similarity to the primary task, may demand the same cognitive structures and, thus, raise the cognitive effort (e.g., Li et al., 2012; Baddeley et al., 1998). Accordingly, it would be interesting to conduct future studies that quantify such a disruptive effect of a concurrent task on the analysts' working memory and to compare the effectiveness of such interruptive obfuscation tasks. Furthermore, the development of obfuscation tasks that could both overload working memory and confuse experts would be an interesting task of future research.

8 Conclusion

Prior to our research, little prior work existed that investigated the underlying human factors in HRE. To address this research gap and generate new insights in this field, I pursued an overarching research goal in my dissertation: exploring problem-solving processes in HRE. Two sub-goals for my dissertation were derived from existing psychological literature on human problem solving. First, I aimed to investigate a potential correlation between HRE problem solving and the level of expertise. Second, I also focused on whether there was a correlation between cognitive abilities (e.g., intelligence or subfactors of intelligence) and HRE problem solving. In conclusion, our research contributed to both the unexplored problem-solving processes in HRE and the correlation of HRE problem solving to cognitive abilities and expertise. In the following, I will summarize our main findings and outline my conclusions.

In order to conduct our main study on HRE problem solving, we first had to circumvent a methodological problem. The methodological problem was that HRE experts were not available for research. As a necessary step, we had to develop an HRE course that promoted the acquisition of HRE skills in students with relevant backgrounds in cyber security. In order to achieve this, we first derived guidelines from existing psychological literature that served as the basis for course development. In an evaluation, we were able to show that the HRE course appeared to promote the acquisition of HRE skills and, consequently, qualified students as participants in our main study.

In order to accomplish the overarching research goal, we conducted an empirical study with eight intermediates and one HRE expert who were asked to solve a realistic HRE task. In order to analyze HRE problem-solving processes, we used an iterative open coding scheme based on Grounded Theory (Strauss & Corbin, 1998) that is known as a well-established methodological approach from the social sciences. Our analysis included 103 unique open codes that were assigned to the problem solving of each participant. Based on the iterative open coding procedure, we developed a detailed hierarchical HRE problem-solving model that comprised the participants' problem-solving processes during HRE. In order to generate in-depth insights into applied problem-solving strategies and their efficiency, we included solution times in our analysis. Our results suggested that two intermediates solved the HRE task by applying different problem-solving strategies than the HRE expert (and each other), but with comparable quickness. Furthermore, we found that some problem-solving strategies were more efficient and others less, and that none of the observed strategies could be described as the single best for the HRE task. Additionally, we found that participants with higher scores in the intelligence sub-factor working memory tended to achieve faster solutions than participants with lower working-memory scores.

Based on our findings, I drew two main conclusions. First, I assumed that HRE problem solving may be a function of both expertise and cognitive abilities. Our findings were in line with prior research on expertise and problem solving. The HRE expert's superior performance was based on efficient problem categorization and representation that influenced the selection of suitable problem-solving strategies. Furthermore, our findings were in line with the expert-performance approach (Ericsson & Kintsch, 1995), suggesting that non-experts can compensate a lack of domain-specific knowledge and problem-solving experience with their cognitive abilities. Against this background, I assumed that both high-performing HRE intermediates (P8, P3) achieved time-efficient solutions due to efficient information processing systems and working memory. I outlined explanations on how an efficient working memory may have supported the analysts in achieving efficient solutions (e.g., simultaneously storing and manipulating netlist information without losing the golden threat of HRE; inhibition of irrelevant netlist components). In this context, I concluded that HRE problems may involve aspects of both simple and complex problems. Although HRE and simple problems have several core aspects in common (e.g., unambiguous initial state; the goal state; no dynamics), the huge amount of information pertaining to non-transparent, highly interconnected components that must be processed and analyzed, creates a challenge that places HRE squarely outside of the realm of simple problems as typically defined and constructed. These characteristics add another layer to HRE problems and thus places them between simple problems and complex problems.

Second, our results were aligned with the considerations by Lee and Johnson-Laird (2013) who suggested that reverse engineering of Boolean systems may be a specific but so far poorly understood type of human problem solving. In this context, I discussed similarities and differences between our findings on HRE problem solving and the results by Lee and Johnson-Laird (2013) on strategies and difficulties during reverse engineering of Boolean systems. For example, we did not find any main strategies in solving the HRE task, but rather individual approaches. However, we agree with Lee's assumption that the difficulty in HRE, like the difficulty in reverse engineering of Boolean systems, could be determined by the number of netlist components and their dependencies.

As the focus of my dissertation lay on the negative and malicious consequences of HRE (i.e., IP theft), I developed initial ideas for cognitive obfuscation that were based on one of my main conclusions that HRE may be a function of both cognitive abilities and expertise. Based on this conclusion, I suggested that cognitive obfuscation tasks should combine cognitively challenging tasks that impede problem-solving processes based on both the cognitive abilities and on expertise. For example, I presented initial considerations on how to impede the support by cognitive abilities, such as the working memory, during HRE. Therefore, I referred to existing psychological and HCI literature on disruptive effects of similar tasks and interruptions (e.g., Li et al., 2012) that could lead to an overload of hardware reverse engineers' working memory. Furthermore, I also developed ideas on how to impede analysts' HRE problem solving that was based on their well-structured domain-specific knowledge and thus, may enable them to circumvent "normal" limits of the working memory (based on long-term working memory as suggested by Ericsson and Kintsch, 1995). Therefore, I referred to prior findings that suggested circumstances and task characteristics through which experts and high-knowledge individuals may be negatively affected in solving the HRE problem efficiently (e.g., Chi, 2006). My presented ideas on cognitive obfuscation should be understood as an impulse for further research. First, it is essential to investigate whether cognitive abilities and expertise influence problem-solving performance in HRE. If future research can successfully quantify this influence, further consideration can be made on how obfuscation tasks might be designed in order to impede problem-solving processes determined by both cognitive abilities and expertise.

Overall, the presented research built a bridge between two disciplines – cognitive psychology and hardware security. By merging both worlds and overcoming disciplinary boundaries, it was possible to achieve the established research goals and generate a deeper understanding of problem-solving processes in HRE. Besides this point, the research also presented an interesting change of perspective in HCI research. In contrast to the common goal of HCI research that is to improve the application and work with certain technologies for specific user groups, we aim to apply our knowledge of problem-solving processes in HRE to impede the analysis of a gate-level netlist.

9 Appendix

9.1 References

Abdel-Hamid, A. T., Tahar, S., & Aboulhamid, E. M. (2003). IP watermarking techniques: Survey and comparison. *The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, 2003. Proceedings.* (pp. 60-65). IEEE.

Acar, Y., Stransky, C., Wermke, D., Mazurek, M. L., & Fahl, S. (2017). Security developer studies with github users: Exploring a convenience sample. *Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)*, (pp. 81-95).

Ackerman, P. L. (1987). Individual differences in skill learning: An integration of psychometric and information processing perspectives. *Psychological bulletin, 102*(1), 3-27.

Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory & cognition, 9*(4), 422-433.

Ainsworth, S. (2006). DeFT: A conceptual framework for considering learning with multiple representations. *Learning and instruction, 16*(3), 183-198.

Ainsworth, S. (2014). 20—The Multiple Representation Principle in Multimedia Learning. *The Cambridge handbook of multimedia learning*, 464.

Airey, J., & Linder, C. (2009). A disciplinary discourse perspective on university science learning: achieving fluency in a critical constellation of modes. *Journal of Research in Science Teaching*, 27-49.

Albartus, N., Hoffmann, M., Temme, S., Azriel, L., & Paar, C. (2020). DANA Universal Dataflow Analysis for Gate-Level Netlist Reverse Engineering. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (pp. 309-336).

Alexander, P. A. (2003). Can we get there from? *Educational Researcher, 32*(8), 3-4.

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive science, 26*(1), 39-83.

Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010). *How learning works: Seven research-based principles for smart teaching.* John Wiley & Sons.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological review, 89*(4), 369.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. . *Psychological Review, 94*, 192-210.

Anderson, J. R. (1993). *Rules of mind.* Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Lebiere, C. (1998). *Atomic components of thought.* Mahwah, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*, 1036-1060.

Azriel, L., Ginosar, R., & Mendelson, A. (2019). SoK: An Overview of Algorithmic Methods in IC Reverse Engineering. *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop,* (pp. 65-74).

Baddeley, A. (1992). Working memory. *Science, 255*(5044), 556-559.

Baddeley, A. (2012). Working memory: theories, models, and controversies. *Annual review of psychology, 63*, 1-29.

Baddeley, A. D. (2000). The episodic buffer: a new component of working memory? *Trends in cognitive sciences, 4*(11), 417-423.

Baddeley, A. D. (2002). Is working memory still working? *European psychologist, 7*(2), 85.

Baddeley, A. D. (2003). Working memory and language: An overview. *Journal of communication disorders, 36*(3), 189-208.

Baddeley, A. D. (2012). Working memory: theories, models, and controversies. *Annual review of psychology, 63,* 1-29.

Baddeley, A. D., & Hitch, G. (1974). Working memory. *Psychology of learning and motivation, 8,* 47-89.

Baddeley, A. D., & Hitch, G. J. (1994). Developments in the concept of working memory. *Neuropsychology, 8*(4), 485.

Baddeley, A., Emslie, H., Kolodny, J., & Duncan, J. (1998). Random generation and the executive control of working memory. *Quarterly Journal of Experimental Psychology, 51A,* 819-852.

Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., & Yang, K. (2001). On the (Im)possibility of Obfuscating Programs. *Annual International Cryptology Conference* (pp. 1-18). Springer.

Becker, G. T., Regazzoni, F., Paar, C., & Burleson, W. P. (2013). Stealthy dopant-level hardware trojans. *International Conference on Cryptographic Hardware and Embedded Systems* (pp. 197-214). Springer, Berlin, Heidelberg.

Becker, S., Wiesen, C., Albartus, N., Rummel, N., & Paar, C. (2020). An Exploratory Study of Hardware Reverse Engineering - Technical and Cognitive Processes. *Proceedings of the Sixteenth Symposium on Usable Privacy and Security ({SOUPS} 2020)* (pp. 285--300). Virtual Conference: USENIX Association.

Bhunia, S., Hsiao, M. S., Banga, M., & Narasimhan, S. (2014). Hardware Trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE, 102*(8), 1229-1247.

Bilalić, M., McLeod, P., & Gobet, F. (2007). Does chess need intelligence?—A study with young chess players. *Intelligence, 35*(5), 457-470.

Bratfisch, O., Borg, G., & Dornic, S. (1972). *Perceived Item-difficulty in Three Tests of Intellectual Performance Capacity.* University of Stockholm, Reports from the Institute of Applied Psychology. Eric.

Brehmer, B. (1992). Dynamic decision making: Human control of complex systems. *Acta Psychologica, 81,* 211-241.

Cades, D. M., Boehm-Davis, D. A., Trafton, J. G., & Monk, C. A. (2007). Does the difficulty of an interruption affect our ability to resume? *Proceedings of the human factors and ergonomics society annual meeting. 51*, pp. 234-238. Sage CA: Los Angeles: CA: SAGE Publications.

Cattell, R. B. (1987). *Intelligence: Its structure, growth and action.* Elsevier.

Ceccato, M., Di Penta, M., Falcarin, P., Ricca, F., Torchiano, M., & Tonella, P. (2014). A family of experiments to assess the effectiveness and efficiency of source code obfuscation techniques. *Empirical Software Engineering*, 1040-1074.

Ceccato, M., Di Penta, M., Nagra, J., Falcarin, P., Ricca, F., Torchiano, M., & Tonella, P. (2009). The effectiveness of source code obfuscation: An experimental assessment. *2009 IEEE 17th International Conference on Program Comprehension* (pp. 178-187). IEEE.

Cellier, J., & Eyrolle, H. (1992). Interference between switched tasks. *Ergonomics, 35*, 25-36.

Chair for Embedded Security. (2019). Retrieved 03 23, 2021, from GitHub: https://github.com/emsec/hal

Chakraborty, R. S., & Bhunia, S. (2009). HARPOON: an obfuscation-based SoC design methodology for hardware protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 28*(10), 1493-1502.

Charmaz, K., & Belgrave, L. L. (2007). Grounded theory. (W. O. Library, Ed.) *The Blackwell encyclopedia of sociology*.

Charness, N., & Tuffiash, M. (2008). The role of expertise research and human factors in capturing, explaining, and producing superior performance. *Human factors, 50*(3), 427-432.

Chase, W. G., & Simon, H. A. (1973a). The mind's eye in chess. In W. G. Chass, *Visual Information Processing* (pp. 215-281). New York: Academic Press.

Chase, W. G., & Simon, H. A. (1973b). Perception in Chess. *Cognitive Psychology, 4*(1), 55-81.

Chen, S. (2019, Juli 11). *Could Huawei be using Trojan circuits to help Beijing spy on the US?* Retrieved October 13, 2020, from South China Morning Post: https://www.techinasia.com/huawei-trojan-circuits-beijing-spy

Chi, M. T. (2006). Two approaches to the study of experts’ characteristics. In K. A. Ericsson, N. Charness, P. J. Feltovich, & R. R. Hoffman, *The Cambridge Handbook of Expertise and Expert Performance* (pp. 21-30). New York: Cambridge University Press.

Chi, M. T., & Ohlsson, S. (2005). Complex declarative learning. In K. J. Holyoak, & R. G. Morrison, *The Cambridge Handbook* (pp. 371-399). Cambridge: Cambridge.

Chi, M. T., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive science, 13*(2), 145-182.

Chi, M. T., Feltovich, P. J., & Glaser, R. (1981a). Categorization and representation of physics problems by experts and novices. *Cognitive science, 5*(2), 121-152.

Chi, M. T., Glaser, R., & Rees, E. (1981b). *Expertise in problem solving.* Pittsburgh Univ PA Learning Research and Development Center.

Chisholm, G. H., Eckmann, S. T., Lain, C. M., & Veroff, R. L. (1999). Understanding Integrated Circuits. *IEEE Design & Test of Computers, 16*(2), 26-37.

Cowan, N. (2005). *Working Memory Capacity* (Vol. 1). New York: Psychology Press.

Crandall, B., & Calderwood, R. (1989). Clinical assessment skills of experienced neonatal intensive care nurses. *Final report, Klein Associates Inc., OH. Prepared under contract, 1,* R43.

Crane, C. (2021, February 17). *Re-Hashed: 27 Surprising IoT Statistics You Don't Already Know*. Retrieved 03 01, 2021, from https://www.thesslstore.com/blog/20-surprising-iot-statistics-you-dont-already-know/

Cutrell, E., Czerwinski, M., & Horvitz, E. (2001). Notification, disruption and memory: Effects of messaging interruptions on memory and performance. *Human-Computer Interaction–Interact '01,* 263-269.

Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior, 19,* 450-433.

De Groot, A. D. (1978). Thought and choice in chess (Vol. 4). *Mouton De Gruyter*.

De Jong, T., Ainsworth, S., Dobson, M., van der Hulst, A., Levonen, J., Reimann, P., . . . Swaak, J. (1998). Acquiring knowledge in science and mathematics: the use of multiple representations in technology based learning environments. *Learning with multiple representations*, 9-40.

Deary, I. J. (2001). Human intelligence differences: Towards a combined experimental-differential approach. *Trends in cognitive sciences, 8*(4), 164-170.

Delaney, P. F. (2018). The Role of Long-Term Working Memory and Template Theory in Contemporary Expertise Research. *Journal of Expertise, 1*(3), 155-161.

Diamond, A. (2013). Executive functions. *Annual review of psychology, 64,* 135-168.

Ding, Z., Wu, Q., Zhang, Y., & Zhu, L. (2013). Deriving an NCD file from an FPGA bitstream: Methodology, architecture and evaluation. *Microprocessors and Microsystems, 37*(3), 299-312.

Dörner, D. (1980). On the difficulties people have in dealing with complexity. *Simulation & Games*, *11*(1), 87-106.

Dörner, D., & Funke, J. (2017). Complex Problem Solving: What It Is and What It Is Not. *Frontiers in Psychology, 8,* 1153.

Dougherty, D. S., & Drumheller, K. (2006). Sensemaking and emotions in organizations: Accounting for emotions in a rational (ized) context. *Communication Studies, 57*(2), 215-238.

Edwards, B. M., & Gronlund, S. D. (1998). Task interruption and its effects on memory. *Memory, 6*(6), 665-687.

Eilam, E. (2011). *Reversing: secrets of reverse engineering.* John Wiley & Sons.

Ender, M., Moradi, A., & Paar, C. (2020). The unpatchable silicon: A full break of the bitstream encryption of xilinx 7-series fpgas. *29th {USENIX} Security Symposium ({USENIX} Security 20),* (pp. 1803-1819).

Ericsson, K. A. (2014). *The road to excellence: The acquisition of expert performance in the arts and sciences, sports, and games.* Psychology Press.

Ericsson, K. A., & Kintsch, W. (1995). Long-term working memory. *Psychological review, 102*(2), 211.

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological review, 100*(3), 363.

Fischer, A., Greiff, S., & Funke, J. (2011). The Process of Solving Complex Problems. *Journal of Problem Solving, 4*(1), 19--42.

Fitts, P. M., & Posner, M. I. (1967). Human performance.

Frensch, P. A., & Sternberg, R. J. (1989). Expertise and intelligent thinking: When is it worse to know better? In R. J. Sternberg, *Advances in the psychology of human intelligence* (pp. 157-188). Hillsdale, NJ: Erlbaum.

Frydman, M., & Lynn, R. (1992). The general intelligence and spatial abilities of gifted young Belgian chess players. *British journal of Psychology, 83*(2), 233-235.

Funke, J. (2003). *Problemlösendes Denken.* Kohlhammer Verlag.

Funke, J. (2012). Complex problem solving. *Encyclopedia of the Sciences of Learning (682-685). Heidelberg: Springer*.

Funke, J., & Frensch, P. A. (2007). Complex problem solving: The European perspective — 10 years after. In D. H. Jonassen, *Learning to solve complex scientific problems.* Lawrence Erlbaum Associates.

Funke, J., Fischer, A., & Holt, D. V. (2018). Competencies for complexity: problem solving in the twenty-first century. In E. Care, P. Griffin, & M. Wilson, *Assessment and teaching of 21st century skills* (1st ed. 2018 Edition ed., pp. 41-53). Springer.

Fyrbiak, M., Strauss, S., Kison, C., Wallat, S., Elson, M., Rummel, N., & Paar, C. (2017). Hardware Reverse Engineering: Overview and Open Challenges. *2017 IEEE 2nd International Verification and Security Workshop* (pp. 88--94). Thessaloniki: IEEE.

Fyrbiak, M., Wallat, S., Déchelotte, J., Albartus, N., Böcker, S., Tessier, R., & Paar, C. (2018b). On the difficulty of fsm-based hardware obfuscation. *IACR Transactions on Cryptographic Hardware and Embedded Systems,* (pp. 293-330).

Fyrbiak, M., Wallat, S., Swierczynski, P., Hoffmann, M., Hoppach, S., Wilhelm, M., & ... Paar, C. (2018a). HAL—The missing piece of the puzzle for hardware reverse engineering, Trojan detection and insertion. *IEEE Transactions on Dependable and Secure Computing, 16*(3), 498-510.

Galton, F. (1870). *Hereditary genius: An inquiry into its laws and consequences.* D. Appleton.

Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware analysis and classification: A survey. . *Journal of Information Security, 5*, 56-64.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive science, 7*(2), 155-170.

Gibson, E. J. (1969). Principles of perceptual learning and development.

Gibson, E. J. (2000). Perceptual learning in development: Some basic concepts. *Ecological Psychology, 12*(4), 295-302.

Gilbert, J. K. (2005). Visualization: A metacognitive skill in science and science education. *Visualization in science education*, 9-27.

Gilbert, J. K. (2008). Visualization: An emergent field of practice and enquiry in science education. *Visualization: Theory and practice in science education*, 3-24.

Gillie, T., & Broadbent, D. E. (1989). What makes interruptions disruptive? A study of length, similarity, and complexity. *Psychological Research, 50*, 243-250.

Gobet, F. (2005). Chunking models of expertise: Implications for education. *Applied Cognitive Psychology, 19*(2), 183-204.

Gobet, F., & Simon, H. A. (1996). Templates in chess memory: A mechanism for recalling several boards. *Cognitive psychology, 31*(1), 1-40.

Gobet, F., & Simon, H. A. (1996). Templates in chess memory: A mechanism for recalling several boards. . *Cognitive psychology, 31*(1), 1-40.

Goldstone, R. (1997). *Perceptual learning.* San Diego, CA: Academic Press.

Gomulkiewicz, R. W., & Williamson, M. L. (1996). *The Problem of Reverse Engineering.*

Gonzalez, C., Thomas, R. P., & Vanyukov, P. (2005). The relationships between cognitive ability and dynamic decision making. *Intelligence, 33*, 169-186.

Gordon, S. (2000). Virus writers: the end of the innocence? *10th Annual Virus Bulletin Conference (VB2000).* Orlando, FL.

Gottfredson, L. S. (1997). Mainstream Science on Intelligence: An Editorial With 52 Signatories, History, and Bibliography . *Intelligence , 24*(1), 13-23.

Grabner, R. H., Neubauer, A. C., & Stern, E. (2006). Superior performance and neural efficiency: The impact of intelligence and expertise. *Brain research bulletin*, 422-439.

Groen, G. J., & Patel, V. L. (1988). The relationship between comprehension and reasoning in medical expertise. In M. T. Chi, R. Glaser, & M. J. Farr, *The nature of expertise* (pp. 287–310). Lawrence Erlbaum Associates, Inc. .

Guin, U., DiMase, D., & Tehranipoor, M. (2014b). Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead. *Journal of Electronic Testing, 30*(1), 9-23.

Guin, U., Huang, K., DiMase, D., Carulli, J. M., Tehranipoor, M., & Makris, Y. (2014a). Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. *102*(8), 1207-1228.

Hambrick, D. Z., Burgoyne, A. P., & Oswald, F. L. (2019). Domain-general models of expertise: The role of cognitive ability. In P. Ward, J. M. Schraagen, J. Gore, & E. Roth, *The Oxford Handbook of Expertise.* Oxford University Press.

Hansen, M. C., Yalcin, H., & Hayes, J. P. (1999). Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test of Computers, 16*(3), 72-80.

Hasher, L., & Zacks, R. T. (1988). Working memory, comprehension, and aging: A review and a new view. *Psychology of learning and motivation, 22*, 193-225.

Jang-Jaccard, J., & Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences, 80*(5), 973-993.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood, M. E. (1981). The processes involved in designing software. *Cognitive skills and their acquisition, 255*, 283.

Joslyn, S., & Hunt, E. (1998). Evaluating individual differences in response to time-pressure. *Journal of Experimental Psychology: Applied*, 16-43.

Kalyuga, S., & Singh, A. M. (2015). Rethinking the boundaries of cognitive load theory in complex learning. *Educational Psychology Review*, 1-22.

Kellman, P. J., & Massey, C. M. (2013). Perceptual learning, cognition, and expertise. *Psychology of learning and motivation, 58*, 117-165.

Kilpatrick, J., Swafford, J., & Findell, B. (2001). The Strands of Mathematical Proficiency. National Research Council. In *Adding It Up: Helping Children Learn Mathematics.* (pp. 115-155). Washington, DC: The National Academies Press.

Kluwe, R. H., Misiak, C., & Haider, H. (1991). The control of complex systems and performance in intelligence tests. In H. Rowe, *Intelligence: reconceptualization and measurement* (pp. 227-244). Hillsdale: Lawrence Erlbaum.

Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., . . . al., e. (2019). Spectre attacks: Exploiting speculative execution. *2019 IEEE Symposium on Security and Privacy (SP)* (pp. 1-19). IEEE.

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science, 36*(5), 757-798.

Kozma, R., & Russell, J. (2005). Students becoming chemists: developing representationl competence. In J. Gilbert, *Visualization in science education* (pp. 121-145). Dordrecht, Netherlands: Springer.

Kozma, R., Chin, E., Russell, J., & Marx, N. (2000). The roles of representations and tools in the chemistry laboratory and their implications for chemistry learning. *The Journal of the Learning Sciences, 9*(2), 105-143.

Kyllonen, P. C., & Woltz, D. J. (1989). Role of cognitive factors in the acquisition of cognitive skill. *Abilities, motivation, and methodology: The Minnesota symposium on learning and individual differences* (pp. 239-280). Hillsdale, NJ: Erlbaum.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980a). Models of competence in solving physics problems. *Cognitive science, 4*(4), 317-345.

Larkin, J., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science*, *208*(4450), 1335-1342.

Lee, N. L., & Johnson-Laird, P. (2013). A theory of reverse engineering and its application to Boolean systems. *Journal of Cognitive Psychology, 25*(4), 365-389.

Lemaire, P., & Siegler, R. S. (1995). Four aspects of strategic change: Contributions to children’s learning of multiplication. *Journal of Experimental Psychology: General, 124*, 83-97.

Lesgold, A., Rubinson, H., Feltovich, P., Glaser, R., Klopfer, D., & Wang, Y. (1988). Expertise in a complex skill: Diagnosing x-ray pictures. In M. T. Chi, R. Glaser, & M. J. Farr, *The nature of expertise* (pp. 311–342). Lawrence Erlbaum Associates, Inc. .

Lexico. (2021). *Lexico.com*. Retrieved 02 04, 2021, from https://www.lexico.com/definition/expert

Li, S. Y., Magrabi, F., & Coiera, E. (2012). A systematic review of the psychological literature on interruption and its patient safety implications. *Journal of the American Medical Informatics Association, 19*(1), 6-12.

Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., . . . al., e. (2018). Meltdown: Reading kernel memory from user space. *27th {USENIX} Security Symposium ({USENIX} Security* (pp. 973-990). USENIX.

Litzinger, T., Lattuca, L. R., Hadgraft, R., & Newstetter, W. (2011). Engineering education and the development of expertise. . *Journal of Engineering Education, 100*(1), 123-150.

Lombard, M., Snyder-Duch, J., & Bracken, C. C. (2002). Content analysis in mass communication: Assessment and reporting of intercoder reliability. *Human communication research, 28*(4), 587-604.

MacAskill, E., & Dance, G. (2013, November 1). *NSA Files: Decoded. What the revelations mean for you.* Retrieved February 26, 2020, from The Guardian: https://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded

Marchant, G., Robinson, J., Anderson, U., & Schadewald, M. (1991). Analogical transfer and expertise in legal reasoning. *Organizational Behavior and Human Decision Making, 48*, 272-290.

Marshall, S. P. (1995). *Schemas in problem solving.* Cambridge University Press.

Mayer, H., Hazotte, C., Djaghloul, Y., Latour, T., Sonnleitner, P., Brunner, M., & Martin, R. (2013). Using complex problem solving simulations for general cognitive ability assessment: The Genetics Lab framework. *International Journal of Information Science and Intelligent System, 2*(4), 71-88.

Mayer, R. E. (1992). *Thinking, problem solving, cognition* (Vol. 2nd ed.). WH Freeman/Times Books/Henry Holt & Co.

Mayer, R. E., & Feldon, D. (2014). Five common but questionable principles of multimedia learning. In R. E. Mayer, *The Cambridge handbook of multimedia learning* (pp. 97-116). New York, NY: Cambridge University Press.

Mayer, R. E., & Wittrock, M. C. (2006). Problem Solving. In P. A. Alexander, & P. H. Winne, *Handbook of Educational Psychology* (pp. 287-303). Mahwah: NJ: Lawrence Erlbaum.

McElhaney, K. W., Chang, H. Y., Chiu, J. L., & Linn, M. C. (2015). Evidence for effective uses of dynamic visualisations in science curriculum materials. *Studies in Science Education, 51*(1), 49-85.

McKeithen, K. B., Reitman, J. S., Reuter, H. H., & Hirtle, S. C. (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology, 13*, 307-325.

Mednick, S. (1962). The associative basis of the creative process. *Psychological Review, 69,* 220-232.

Michaels, S., O'Connor, C., & Resnick, L. B. (2008). Deliberative discourse idealized and realized: Accountable talk in the classroom and in civic life. *Studies in philosophy and education, 27*(4), 283-297.

Michalchik, V., Rosenquist, A., Kozma, R., Kreikemeier, P., & Schank, P. (2008). Representational resources for constructing shared understandings in the high school chemistry classroom. *Visualization: Theory and practice in science education*, 233-282.

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review, 63*(2), 81-97.

Mizoguchi, K., Yuzurihara, M., Ishige, A., Sasaki, H., Chui, D. H., & Tabira, T. (2000). Chronic stress induces impairment of spatial working memory because of prefrontal dopaminergic dysfunction. *Journal of Neuroscience, 20*(4), 1568-1574.

Monk, C. A., Trafton, J. G., & Boehm-Davis, D. A. (2008). The effect of interruption duration and demand on resuming suspended goals. *Journal of experimental psychology: Applied, 14*(4), 299.

Moradi, A., Barenghi, A., Kasper, T., & Paar, C. (2011). On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs., (pp. 111-124). Proceedings of the 18th ACM conference on Computer and communications security.

Moran, T. P. (2016). Anxiety and working memory capacity: A meta-analysis and narrative review. *Psychological Bulletin, 142*(8), 831.

Moss, J., Kotovsky, K., & Cagan, J. (2006). The role of functionality in the mental representations of engineering students: Some differences in the early stages of expertise. *Cognitive Science, 30*(1), 65-93.

Naiakshina, A., Danilova, A., Gerlitz, E., & Smith, M. (2020). On conducting security developer studies with cs students: Examining a password-storage study with cs students, freelancers, and company developers. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* , (pp. 1-13).

Naiakshina, A., Danilova, A., Gerlitz, E., von Zezschwitz, E., & Smith, M. (2019). " If you want, I can store the encrypted password" A Password-Storage Field Study with Freelance Developers. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems,* (pp. 1-12).

Naiakshina, A., Danilova, A., Tiefenau, C., & Smith, M. (2018). Deception task design in developer password studies: Exploring a student sample. *Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018),* (pp. 297-313).

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving (Vol. 104, No. 9).* Englewood Cliffs: NJ: Prentice-Hall.

Nokes, T. J., Schunn, C. D., & Chi, M. (2010). Problem solving and human expertise. In *International encyclopedia of education* (pp. 265-272). Elsevier Ltd.

Norman, G. R., Brooks, L. R., & Allen, S. W. (1989). Recall by expert medical practitioners and novices as a record of processing attention. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 15,* 1166-1174.

Note, J. B., & Rannaud, É. (2008). From the bitstream to the netlist. *FPGA, Vol. 8,* 264-264.

Novick, L. R., & Bassok, M. (2005). Problem solving. In K. J. Holyoak, & R. G. Morrison, *The Cambridge handbook of thinking and reasoning* (pp. 321-349). Cambridge: NY: University Press.

Oulasvirta, A., & Saariluoma, P. (2004). Long-term working memory and interrupting messages in human–computer interaction. *Behaviour & Information Technology, 23*(1), 53-64.

Paas, F. G. (1992). Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of educational psychology, 84*(4), 429.

Pashler, H., Bain, P. M., Bottge, B. A., Graesser, A., Koedinger, K., McDaniel, M., & Metcalfe, J. (2007). Organizing Instruction and Study to Improve Student Learning. IES Practice Guide. NCER 2007-2004. *National Center for Education Research.*

Patel, V. L., & Groen, G. J. (1986). Knowledge based solution strategies in medical reasoning. *Cognitive science, 10*(1), 91-116.

Patel, V. L., & Kaufman, D. R. (1995). Clinical reasoning and biomedical knowledge: implimplications for teaching. In J. Higgs, & M. Jones, *Clinical reasoning in the health professions* (pp. 117-128). Oxford, UK: Butterworth-Heinemann Ltd.

Pecht, M., & Tiku, S. (2006). Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics. *IEEE spectrum, 43*(5), 37-46.

Pintrich, P. R. (2003). A motivational science perspective on the role of student motivation in learning and teaching contexts. . *Journal of educational Psychology, 95*(4), 667.

Putz-Osterloh, W. (1985). Selbstreflexionen, Testintelligenz und interindividuelle Unterschiede bei der Bewältigung komplexer Probleme [Self-reflections, test intelligence and interindividual differences in solving complex problems]. *Sprache & Kognition, 4,* 203-216.

Qin, S., Hermans, E. J., van Marle, H. J., Luo, J., & Fernández, G. (2009). Acute psychological stress reduces working memory-related activity in the dorsolateral prefrontal cortex. . *Biological psychiatry, 66*(1), 25-32.

Quadir, S., Chen, J., Forte, D., Asadizanjani, N., Shahbazmohamadi, S., Wang, L., . . . Tehranipoor, M. (2016). A Survey on Chip to System Reverse Engineering. *ACM Journal on Emerging Technologies in Computing Systems, 13*(1), 6.

Rau, M. A. (2017). Conditions for the effectiveness of multiple visual representations in enhancing STEM learning. *Educational Psychology Review, 29*(4), 717-761.

Raven, J. C., Raven, J. C., & De Lemos, M. (1958). *Standard progressive matrices.* London: Lewis.

Reither, F. (1981). Thinking and acting in complex situations: A study of experts' behavior. *12*(2), 125-140.

Reitman, W. R. (1965). Cognition and thought: an information processing approach.

Rekoff, M. G. (1985). On Reverse Engineering. *IEEE Transactions on Systems, Man, and Cybernetics*, 244--252.

Resnick, L., & Glaser, R. (1975). Problem Solving and Intelligence. In L. Resnick, *The nature of intelligence.* Hillsdale: Lawrence Erlbaum.

Rheinberg, F., Vollmeyer, R., & Burns, B. D. (2001). QCM: A questionnaire to assess current motivation in learning situations. *Diagnostica, 47*(2), 57-66.

Richman, H. B., Gobet, F., Staszewski, J. J., & Simon, H. A. (1996). Perceptual and memory processes in the acquisition of expert performance: The EPAM model. In K. A. Ericsson, *The road to excellence: The acquisition of expert performance in the arts and sciences, sports, and games* (pp. 167-187). Lawrence Erlbaum.

Rigas, G., & Brehmer, B. (1999). Mental processes in intelligence tests and dynamics decision making tasks. In P. J. Juslin, & H. Montgomerry, *Judgment and decision making: Neo-Brunswikian and process-tracing approaches* (pp. 45-65). Hillsdale: Lawrence Erlbaum.

Rocke, A. J. (2010). *Image and reality: Kekulé, Kopp, and the scientific imagination.* University of Chicago Press.

Rostami, M., Koushanfar, F., & Karri, R. (2014). A primer on hardware security: Models, methods, and metrics. *Proceedings of the IEEE, 102*(8), 1283-1295.

Sasse, M. A., Brostoff, S., & Weirich, D. (2001). Transforming the 'Weakest Link'—A Human-Computer Interaction Approach for Usable and Effective Security. *BT Technology Journal, 13*(3), 122-131.

Schmeck, A., Opfermann, M., Van Gog, T., Paas, F., & Leutner, D. (2015). Measuring cognitive load with subjective rating scales during problem solving: differences between immediate and delayed ratings. *Instructional Science*, *43*(1), 93-114.

Schmid, M., Ziener, D., & Teich, J. (2008). Netlist-level IP protection by watermarking for LUT-based FPGAs. *2008 International Conference on Field-Programmable Technology* (pp. 209-216). Taipei, Taiwan: IEEE.

Schmidt, F. L., & Hunter, J. (2004). General mental ability in the world of work: occupational attainment and job performance. *Journal of personality and social psychology, 86*(1), 162-173.

Schneider, W. (1985). Training high-performance skills: Fallacies and guidelines. *Human factors*, *27*(3), 285-300.

Schnotz, W. (2014). Integrated Model of Text and Picture Comprehension. In R. E. Mayer, *The Cambridge handbook of multimedia learning* (pp. 72-103). New York, NY: Cambridge University Press.

Schnotz, W., & Bannert, M. (2003). Construction and interference in learning from multiple representation. *Learning and instruction, 13*(2), 141-156.

Schraw, G., Dunkle, M. E., & Bendixen, L. D. (1995). Cognitive processes in well-defined and ill-defined problem solving. *Applied Cognitive Psychology, 9*(6), 523-538.

Seufert, T. (2003). Supporting coherence formation in learning from multiple representations. *Learning and instruction, 13*(2), 227-237.

Shakya, B., Tehranipoor, M. M., Bhunia, S., & Forte, D. (2017). Introduction to hardware obfuscation: Motivation, methods and evaluation. *Hardware Protection through Obfuscation,* 3-32.

Sherin, B. L. (2000). Meta-representation: An introduction. *The Journal of Mathematical Behavior, 19*(4), 385-398.

Silver, E. A. (1979). Student perceptions of relatedness among verbal. *Journal of Research in Mathematics Education, 10*, 195–210.

Sim, J. H., & Daniel, E. G. (2014). Sim, J. H., & Daniel, E. G. S. (2014). Representational competence in chemistry: A comparison between students with different levels of understanding of basic chemical concepts and chemical representations. *Cogent Education, 1*(1).

Simon, D. P., & Simon, H. A. (1978). Individual differences in solving physics problems. In R. S. Siegler, *Children's thinking: What develops?* (pp. 325–348). Lawrence Erlbaum Associates, Inc. .

Simonton, D. K. (1999). Talent and its development: An emergenic and epigenetic model. *Psychological review, 106*(3), 435.

Singer-Dudek, J., & Greer, R. D. (2005). A long-term analysis of the relationship between fluency and the training and maintenance of complex math skills. *The Psychological Record, 55*(3), 361-376.

Skinner, C. H., Fletcher, P. A., & Henington, C. (1996). Increasing learning rates by increasing student response rates: A summary of research. *School Psychology Quarterly, 11*(4), 313.

Spearman, C. E. (1927). *The abilities of man* (Vol. 89). New York: Macmillan.

Stadler, M., Becker, N., Gödker, M., Leutner, D., & Greiff, S. (2015). Complex problem solving and intelligence: A meta-analysis. *Intelligence, 53*, 92-101.

Stephanidis, C., Salvendy, G., Antona, M., Chen, J., Dong, J., Duffy, V., . . . Fu, L. (2019). Seven HCI grand challenges. *International Journal of Human--Computer Interaction, 35*(14), 1229--1269.

Sternberg, R. J. (1982). *Handbook of human intelligence.* CUP Archive.

Sternberg, R. J., & Berg, C. A. (1986). Quantitative integration. Definition of intelligence: A comparison of the 1921 and 1986 symposia. In R. J. Sternberg, & D. K. Detterman, *What is intelligence?* (pp. 155-162). Norwood, NJ: Ablex.

Sternberg, R. J., & Frensch, P. A. (1992). On being an expert;Acost-benefit analysis. In R. R. Hoffman, *The psychology of expertise: Cognitive research and empirical AI* (pp. 191-203). New York: Springer Verlag.

Strauss, A. L., & Corbin, J. M. (1998). *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory.* Sage Publications, Inc.

Subramanyan, P., Tsiskaridze, N., Li, W., Gascón, A., Tan, W. Y., Tiwari, A., & ... Malik, S. (2014). Reverse engineering digital circuits using structural and functional analyses. *IEEE Transactions on Emerging Topics in Computing, 2*(1), 63-80.

Süß, H. M., Kersting, M., & Oberauer, K. (1991). Intelligenz und Wissen als Prädiktoren für Leistungen bei computersimulierten komplexen Problemen [Intelligence and knowledge as predictors of success in computer simulated complex problems]. *Diagnostica, 37*, 334-352.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction, 2*(1), 59-89.

Tarnovsky, C. (2019). *Hack To Discover Weaknesses In A Series Of Smartcards.* Retrieved 03 01, 2021, from https://www.youtube.com/watch?v=2td3-sWsiKg

Tenison, C., & Anderson, J. R. (2016). Modeling the distinct phases of skill acquisition. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 42*(5), 749.

Thomas, O. (2015). *Advanced IC Reverse Engineering Techniques: In Depth Analysis Of A Modern Smart Card.* Retrieved 03 01, 2021, from https://www.youtube.com/watch?v=YM5I8yR7yCw

Thurstone, L. L. (1938). *Primary mental abilities* (Vol. 119). Chicago: University of Chicago Press.

Torrance, R., & James, D. (2009). The state-of-the-art in IC reverse engineering. *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 363-381). Springer, Berlin, Heidelberg.

Trafton, J. G., Altmann, E. M., Brock, D. P., & Mintz, F. E. (2003). Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies, 58*(5), 583-603.

Voss, J. F., & Post, T. A. (1988). On the solving of ill-structured problems. In M. T. Chi, R. Glaser, & M. J. Farr, *The Nature of Expertise* (pp. 261-285). NJ: Erlbaum.

Voss, J. F., Vesonder, G. T., & Spilich, G. J. (1980). Text generation and recall by high-knowledge and low-knowledge individuals. *Journal of verbal Learning and verbal Behavior, 19*(6), 651-667.

Votipka, D., Rabin, S., Micinski, K., Foster, J. S., & Mazurek, M. L. (2020). An Observational Investigation of Reverse Engineers' Processes. *29th {USENIX} Security Symposium ({USENIX} Security 20),* (pp. 1875-1892).

Wallat, S., Albartus, N., Becker, S., Hoffmann, M., Ender, M., Fyrbiak, M., & ... Paar, C. (2019). Highway to HAL: open-sourcing the first extendable gate-level netlist reverse engineering framework. *Proceedings of the 16th ACM International Conference on Computing Frontiers,* (pp. 392-397).

Wallat, S., Fyrbiak, M., Schlögel, M., & Paar, C. (2017). A look at the dark side of hardware reverse engineering-a case study. *2017 IEEE 2nd International Verification and Security Workshop (IVSW)* (pp. 95-100). IEEE.

Wechsler, D. (2008). Wechsler adult intelligence scale–Fourth Edition (WAIS–IV). *San Antonio, TX: NCS Pearson, 22*(498), 1.

Wenke, D., Frensch, P. A., & Funke, J. (2005). Complex problem solving and intelligence: Empirical relation and causal direction. In R. J. Sternberg, & J. E. Pretz, *Cognition and intelligence: Identifying the mechanisms of the mind* (pp. 160-187). Cambridge University Press.

Wertsch, J. V., & Kazak, S. (2011). Saying more than you know in instructional settings. *Theories of learning and studies of instructional practice*, 153-166.

Wertsch, J. V., & Kazak, S. (2011). Saying more than you know in instructional settings. In T. Koschmann, *Theories of learning and studies of instructional practice* (pp. 153-166). New York: Springer.

Weste, N. H., & Harris, D. (2015). CMOS VLSI design: a circuits and systems perspective. (P. E. India., Ed.)

Wiesen, C., Albartus, N., Hoffmann, M., Becker, S., Wallat, S., Fyrbiak, M., Rummel, N., & Paar, C. (2019a). Towards Cognitive Obfuscation: Impeding Hardware Reverse Engineering Based on Psychological Insights. *Proceedings of the 24th Asia and South Pacific Design Automation Conference,* 104-111.

Wiesen, C., Becker, S., Albartus, N., Paar, C., & Rummel, N. (2019b). Promoting the Acquisition of Hardware Reverse Engineering Skills. *2019 IEEE Frontiers in Education Conference (FIE)* (pp. 1-9). Cincinnati (USA): IEEE.

Wiesen, C., Becker, S., Fyrbiak, M., Albartus, N., Elson, M., Rummel, N., & Paar, C. (2018). Teaching Hardware Reverse Engineering: Educational Guidelines and Practical

Insights. *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering* (pp. 438-445). Sydney, Australia: IEEE.

Wiley, J. (1998). Expertise as mental set: The effects of domain knowledge in creative problem solving. *Memory & cognition, 26*(4), 716-730.

Willcutt, E. G., Doyle, A. E., Nigg, J. T., Faraone, S. V., & Pennington, B. F. (2005). Validity of the executive function theory of attention-deficit/hyperactivity disorder: a meta-analytic review. *Biological psychiatry, 57*(11), 1336-1346.

Wineburg, S. (1998). Reading Abraham Lincoln: An expert/expert study in the interpretation of historical texts. *Cognitive science, 22*(3), 319-346.

Wittmann, W. W., & Süß, H. M. (1999). Investigating the paths between working memory,intelligence, knowledge, and complex problem-solving performances via Brunswik symmetry. In R. D. Roberts, *Learning and individual differences: Process, trait, and content determinants* (pp. 77-108). Washington, DC: American Psychological Association.

Wüstenberg, S., Greiff, S., & Funke, J. (2012). Complex problem solving — More than reasoning? *Intelligence, 40*, 1-14.

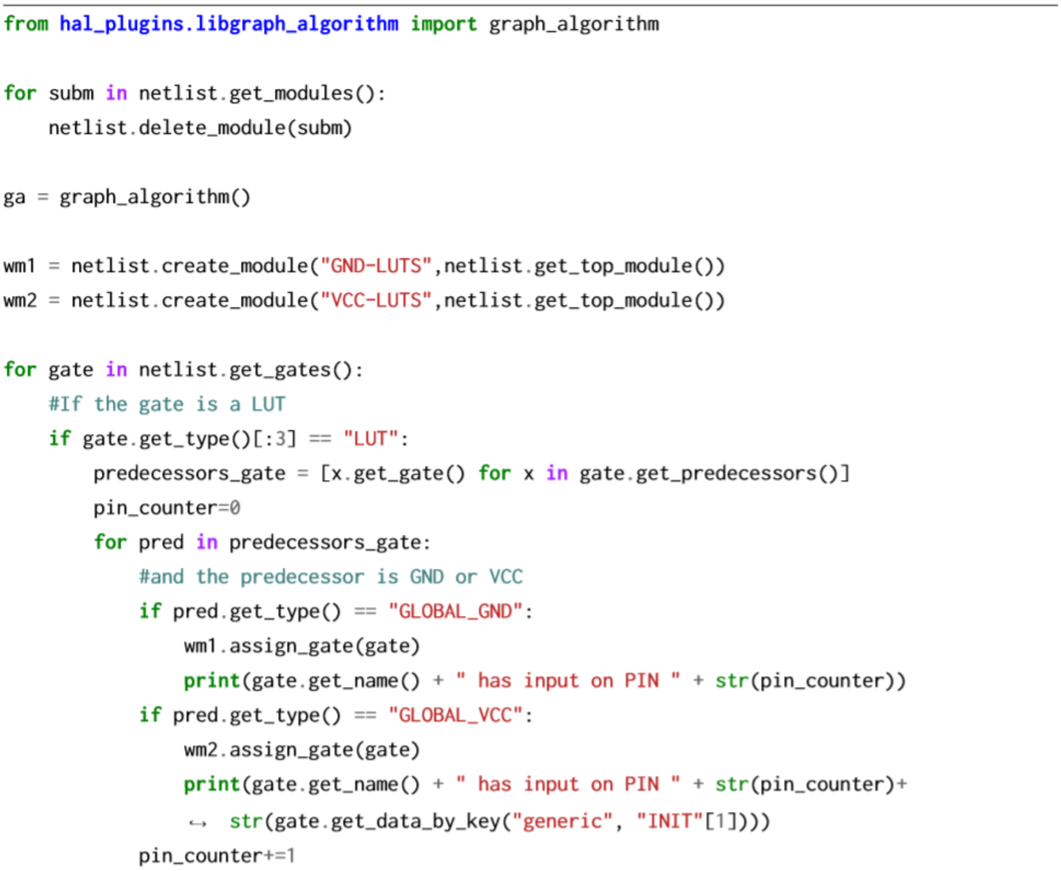
Xilinx, I. (2010, February). *Spartan-6 FPGA Configurable Logic Block.* Retrieved March 20, 2019, from https://www.xilinx.com/support/documentation/user_guides/ug384.pdf

Zacks, & Hasher. (2006). Aging and long-term memory: Deficits are not inevitable. In E. Bialystock, & F. I. Craik, *Lifespan Cognition: Mechanisms of Change* (pp. 168-177). New York: Oxford Univ. Press.

Zijlstra, F. R., Roe, R. A., Leonora, A. B., & Krediet, I. (1999). Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology, 72*(2), 163-185.

9.2 Example Script

Appendix 1



```python
from hal_plugins.libgraph_algorithm import graph_algorithm

for subm in netlist.get_modules():
    netlist.delete_module(subm)

ga = graph_algorithm()

wm1 = netlist.create_module("GND-LUTS",netlist.get_top_module())
wm2 = netlist.create_module("VCC-LUTS",netlist.get_top_module())

for gate in netlist.get_gates():
    #If the gate is a LUT
    if gate.get_type()[:3] == "LUT":
        predecessors_gate = [x.get_gate() for x in gate.get_predecessors()]
        pin_counter=0
        for pred in predecessors_gate:
            #and the predecessor is GND or VCC
            if pred.get_type() == "GLOBAL_GND":
                wm1.assign_gate(gate)
                print(gate.get_name() + " has input on PIN " + str(pin_counter))
            if pred.get_type() == "GLOBAL_VCC":
                wm2.assign_gate(gate)
                print(gate.get_name() + " has input on PIN " + str(pin_counter)+
                ↪  str(gate.get_data_by_key("generic", "INIT"[1])))
            pin_counter+=1
```

Figure 9. Example script executed by Participant 8 while working on the HRE problem-solving task.

9.3 Detailed Code Book

Appendix 2

Here, we list the final codebook used to annotate the content of each log file. The codebook is divided into nine parts that reflect the nine sub-categories of our HRE problem-solving model (see Figure 2). All 103 open codes are assigned to one corresponding sub-category. In the following, we provide a short description for each open code.

Error Introduction.

- *introduction of syntactical errors*: Analyst introduced one or several syntactical errors in the solution script (as indicated by a “SyntaxError” of the interpreter).
- *introduction of repeated syntactical errors*: Analyst repeatedly introduced the very same syntactical error in the solution script.
- *choice of inappropriate technical approaches*: Analyst chose an inappropriate programming approach to solve the problem at hand (e.g., wrong function).
- *introduction of repeated semantic errors*: Analyst repeatedly introduced the very same semantic error in the solution script.

Troubleshooting.

- *attempt to correct syntactical errors*: Analyst unsuccessfully tried to correct one or several existing syntactical errors in the solution script.
- *successful correction of syntactical errors*: Analyst successfully corrected one or several existing syntactical errors in the solution script.
- *attempt to correct semantic errors*: Analyst unsuccessfully tried to correct one or several existing semantic errors in the solution script.
- *successful correction of semantic errors*: Analyst successfully corrected one or several existing semantic errors in the solution script.
- *general debugging*: Analyst attempted to locate a flaw in the solution script.
- *attempt to fix persistent problems*: Analyst located and attempted to correct parts of a persistent error in the solution script.
- *successful correction of persistent problems*: Analyst understood and fully resolved a persistent error in the solution script.
- *workaround*: Analyst located an issue and bypassed it to be able to continue working.
- *attempt to correct a non-existent semantic error*: Analyst assumed a semantic error and attempted to correct it while no such error is in fact present in the solution script.
- *hackfix*: Analyst resolved a syntactical or semantic error by the use of an inefficient or inappropriate but generally functional technique.

Test and Validation.

- *targeted verification*: Analyst checked a small, specific part of the algorithm for some well-defined expected behavior or output.

- *targeted evaluation*: Analyst checked a small, specific part of the algorithm for basic plausibility of its behavior or output.
- *general evaluation*: Analyst tested the behavior of a large section of the algorithm or of the entire algorithm.
- *manual netlist inspection for script validation*: Analyst compared an insight gained from the solution script against information obtained by inspecting parts of the netlist through the graphical user interface.
- *reliability test*: Analyst verified that the algorithm repeatedly and consistently produces the expected result (e. g., after restarting HAL and thus resetting any saved state).
- *testing python constructs*: Analyst tested functionality of the programming language.
- *testing the hal api*: Analyst tested functionality of the HAL scripting system (e. g., functions to access the netlist).

Code Adjustments.

- *improve clarity of console output*: Analyst made the console output more readable and understandable (e. g., by using better formatting or removing unnecessary output).
- *improve clarity of the code*: Analyst made the solution script more readable (e. g., by adding comments or including white spaces).
- *cleaning up within the code*: Analyst removed obsolete parts of the solution script (e. g., temporary testing code).
- *code simplification*: Analyst removed not strictly needed parts from the solution script.
- *code restructuring*: Analyst reorganized or rewrote parts of the solution script to improve its functional structure.
- *code optimization for error avoidance*: Analyst recognized and redesigned an error-prone programming solution, or added error prevention or reporting mechanisms to the solution script.
- *restoration of a functional code state*: Analyst rolled back one or several problematic code changes to restore the last known-working version.
- *reversion to previous code components*: Analyst copied code from a previous version of the solution script into the current version.
- *introduction of redundant code*: Analyst duplicated code with minimal changes, unnecessarily enlarging the solution script.
- *code artifact remains*: Analyst left obsolete pieces of code in the solution script.

- *deletion of actually needed code*: Analyst removed code that other parts of the solution script were dependent on.
- *anticipatory documentation*: Analyst added comments to the solution script as a means of planning their next steps.
- *explanatory documentation*: Analyst added comments documenting the code.
- *documentation of results*: Analyst gathered information required to document their findings (e. g., by printing specific results of the algorithm).

Inspection and Information Gathering.

- *manual netlist exploration*: Analyst navigated through parts of the netlist using the graphical user interface of HAL.
- *script-based netlist exploration*: Analyst used the scripting functionality of HAL to generate statistics or locate interesting parts of the netlist at hand.
- *script-based search for gnd and vcc nets2*: Analyst used the scripting functionality of HAL to search for the Ground (gnd) and Voltage common collector (vcc) nets.
- *script-based identification of the gnd gate*: Analyst successfully located the GND gate using the scripting functionality.
- *script-based identification of the vcc gate*: Analyst successfully located the VCC gate using the scripting functionality.
- *script-based inspection of the gnd and vcc gates*: Analyst used the scripting functionality to access information about the GND and VCC gates.
- *script-based inspection of the input pin types*: Analyst used the scripting functionality to access information about the input pins of a gate.
- *script-based inspection of input nets*: Analyst used the scripting functionality to access information about the nets feeding into the input pins of a gate.
- *script-based inspection of global inout nets*: Analyst used the scripting functionality to access information about the global in-/output nets.
- *script-based inspection of watermark candidates*: Analyst used the scripting functionality to access information about the watermark candidates identified by their algorithm.
- *general script-based netlist inspection*: Analyst used the scripting functionality to access information about interesting parts of the netlist at hand.
- *first manual identification of the gnd net*: Analyst successfully located the GND net by browsing the netlist via the graphical user interface.

- *first manual identification of the vcc net*: Analyst successfully located the VCC net by browsing the netlist via the graphical user interface.
- *first manual identification of the gnd gate*: Analyst successfully located the GND gate by browsing the netlist via the graphical user interface.
- *first manual identification of the vcc gate*: Analyst successfully located the VCC gate by browsing the netlist via the graphical user interface.
- *manual selection of a subsequent gate from gnd or vcc*: Analyst selected one or several gates connected to VCC or GND (i. e., gates possibly relevant for the watermarking) via the graphical user interface.
- *manual selection of irrelevant gates*: The analyst selected one or more gates that were not helpful for watermark recognition via the graphical user interface.
- *manual inspection of irrelevant gates*: Analyst checked details of one or several gates that were not helpful for watermark recognition via the graphical user interface.
- *manual selection of irrelevant nets*: Analyst selected one or several nets that were not helpful for the detection of the watermarking via the graphical user interface.
- *manual selection of watermark candidates*: Analyst selected one or several gates that may contain watermarking via the graphical user interface.
- *manual inspection of the gnd net*: Analyst checked details of the GND net via the graphical user interface.
- *manual inspection of the vcc net*: Analyst checked details of the VCC net via the graphical user interface.
- *manual inspection of the gnd gate*: Analyst checked details of the GND gate via the graphical user interface.
- *manual inspection of the vcc gate*: Analyst checked details of the VCC gate via the graphical user interface.
- in-*depth manual inspection of watermark candidates*: Analyst spent significant time checking details of one or several gates that may contain watermarking via the graphical user interface.

Reversing Strategy Decisions.

- *small-step preparation of a reversing sub*-step: Analyst introduced small local code change or tested functionality to support the next step in the reversing process.
- *preparation of a reversing sub-step*: Analyst introduced significant code change or design decision to support the next step in the reversing process.

- *using external resources*: Analyst used external (online) documentation or performed manual analysis on paper or otherwise outside the HAL environment.
- *generic approach*: Analyst implemented (part of) an algorithm able to work on a multitude of input conditions (i. e., in opposition to writing highly redundant code).
- *recourse to proven approach*: Analyst reused or adapted a known-working strategy.
- *combining individual approaches*: Analyst merges two separately developed parts of the solution script into one algorithm.
- *duplication of partial solutions from gnd to vcc (and vice versa)*: Analyst duplicated parts of the algorithm targeted to GND respective VCC to solve the other case.
- *duplication of partial solutions for watermarking candidates*: Analyst duplicated parts of the algorithm targeted specifically to a subset of candidate gates to solve another subset.
- *fully manual approach*: Analyst refrains from using the scripting functionality to solve the current problem and rather performs the analysis by hand.
- *local approach without considering the overall problem*: Analyst attempted to solve an issue without aligning with the overall goal or checking for negative consequences of their local fix.
- *brute force approach*: Analyst tried to solve a problem by enumerating and testing the entire solution space, rather than using a more efficient approach.
- *hardcoding approach*: Analyst entered information about the netlist at hand (e. g., names of specific gates) manually into the code, restricting its applicability to the current netlist.
- *development of test cases*: Analyst developed a piece of code specifically to test a solution idea separated from the final solution script.
- *selection of test candidates*: Analyst hardcoded (new) candidate gate(s) to execute the test code snippet on them.
- *generalisation of a specific approach*: Analyst modified parts of the algorithm previously targeted to a specific input candidate or subset of candidates to work on a broader input range.
- *renewed solution attempt without change of strateg*y: Analyst re-attempted implementation of some functionality that he or she had given up on before.
- *strategy change for script-based analysis*: Analyst chose a new, alternative approach to solve the problem at hand while using the scripting functionality.

- *strategy change from script-based to manual analysis*: Analyst switched to solving the problem at hand manually, rather than pursuing a working automated solution.
- *strategy change for manual analysis*: Analyst chose a new, alternative approach to solve the problem at hand while solely using the graphical user interface.

Reversing Milestones and Sub-Goals.

- *achieving a milestone: identification of watermarking candidates*: Analyst successfully identified the relevant gates for the subsequent watermarking extraction and removal.
- *achieving a mileston*e: *removal of the watermarking*: Analyst successfully patched the watermarking out of the identified candidate gates.
- achieving a milestone: *extraction of the watermarking*: Analyst successfully read out the watermarking from the identified candidate gates.
- *achieving a milestone:* Analyst completed a key step of their personal solution strategy.
- *successfully completed sub-step for the removal of the watermarking*: Analyst solved a key step in patching the watermarking.
- *successfully completed sub-step for the extraction of the watermarking*: Analyst solved a key step in reading out the watermarking.
- *successfully completed sub-step for the identification of watermarking candidates*: Analyst solved a key step in identifying the candidates relevant for the watermarking.
- *systematic approach with regard to the removal of the watermarking*: Analyst devised a clear, suitable strategy to patch the watermarking and started working towards it.
- *systematic approach with regard to extraction of the watermarking*: Analyst devised a clear, suitable strategy to read out the watermarking and started working towards it.
- *systematic approach with regard to the identification of watermarking candidates*: Analyst devised a clear, suitable strategy to identify the watermarking candidates and started working towards it.
- *successful application of a reversing-specific concept*: Analyst examined and understood a reversing concept and began using it for their solution.

Reversing Problems.

- *reversing-specific lack of understanding*: Analyst misunderstood or missed a central reversing concept and thus, implemented an unsuitable solution for the problem at hand.
- *attempt to correct the reversing-specific misconception*: Analyst located a misconception in the solution script and unsuccessfully attempted to correct it.

- *correction of the reversing-specific misconception*: Analyst understood a reversing concept and subsequently corrected a misconception in the solution script.
- *unsuccessful transfer of an already known approach to a current problem*: Analyst made a mistake while attempting to apply a previously known solution to the current problem.
- *lost track of the reversing approach*: Analyst had lost the overview of their solution approach and was stuck on a local problem.
- *inappropriate change of the reversing strategy*: Analyst switched to another solution strategy while their current approach was already well-suited.
- *correct solution is not recognized*: Analyst solved the problem at hand but did not realize that they did so.
- *dead end*: Analyst gave up after applying a strategy that was inappropriate to the current problem.

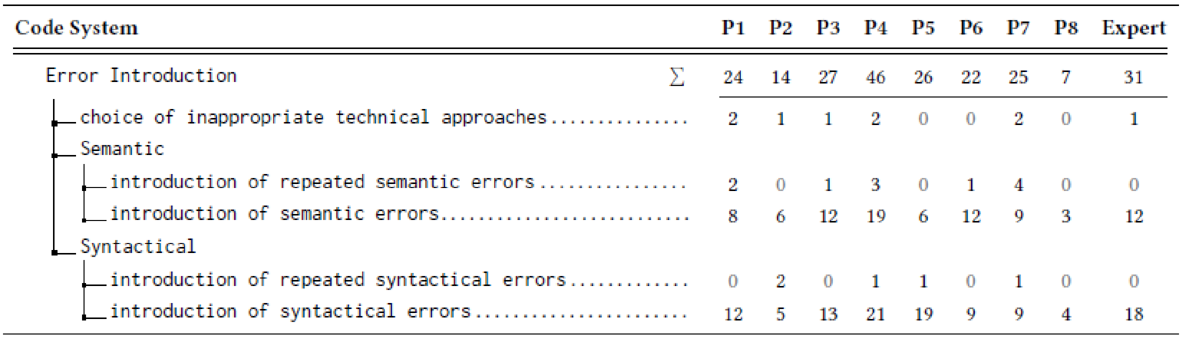
External Influences.

- *software bug*: Analyst‘s progress was hindered by a software error in the HAL environment.
- *unintentional manual selection*: Analyst unintentionally selected an element in the graphical user interface.
- *external interruption*: Analyst‘s work was interrupted by an external influence (e. g., breaks).
- *reinsurance and re-entry after interruption*: Analyst verified the current state of their work after being interrupted by an external influence.

9.4 Full HRE Taxonomy

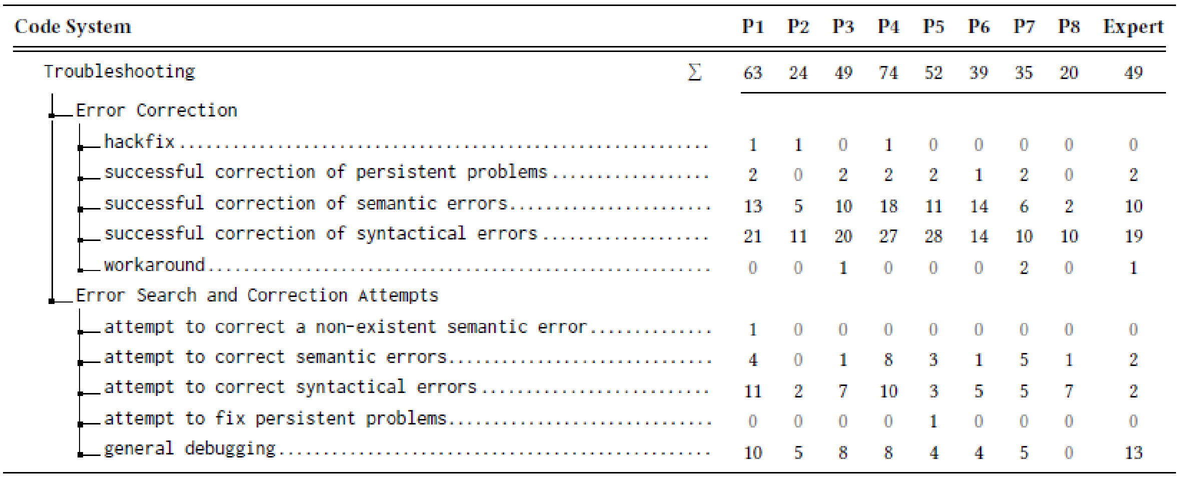
Appendix 3

Table 8. Hierarchy of the sub-category Error Introduction and absolute number of assigned open codes per participant.



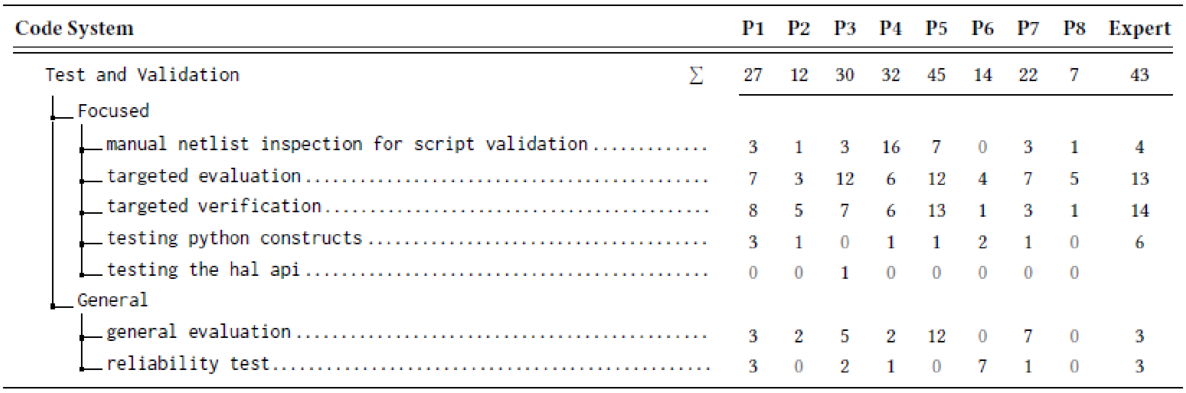
Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Error Introduction	Σ	24	14	27	46	26	22	25	7	31
choice of inappropriate technical approaches		2	1	1	2	0	0	2	0	1
Semantic										
introduction of repeated semantic errors		2	0	1	3	0	1	4	0	0
introduction of semantic errors		8	6	12	19	6	12	9	3	12
Syntactical										
introduction of repeated syntactical errors		0	2	0	1	1	0	1	0	0
introduction of syntactical errors		12	5	13	21	19	9	9	4	18

Table 9. Hierarchy of the sub-category Troubleshooting and absolute number of assigned open codes per participant.



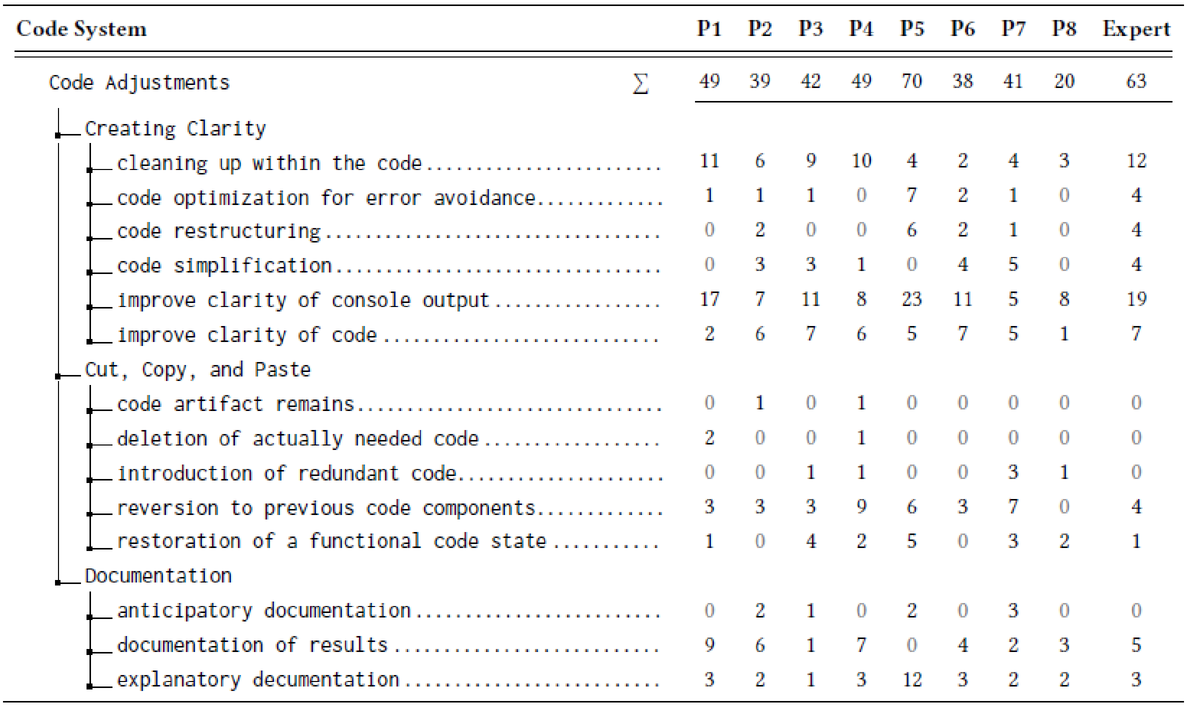
Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Troubleshooting	Σ	63	24	49	74	52	39	35	20	49
Error Correction										
hackfix		1	1	0	1	0	0	0	0	0
successful correction of persistent problems		2	0	2	2	2	1	2	0	2
successful correction of semantic errors		13	5	10	18	11	14	6	2	10
successful correction of syntactical errors		21	11	20	27	28	14	10	10	19
workaround		0	0	1	0	0	0	2	0	1
Error Search and Correction Attempts										
attempt to correct a non-existent semantic error		1	0	0	0	0	0	0	0	0
attempt to correct semantic errors		4	0	1	8	3	1	5	1	2
attempt to correct syntactical errors		11	2	7	10	3	5	5	7	2
attempt to fix persistent problems		0	0	0	0	1	0	0	0	0
general debugging		10	5	8	8	4	4	5	0	13

Table 10. Hierarchy of the sub-category Test and Validation and absolute number of assigned open codes per participant.



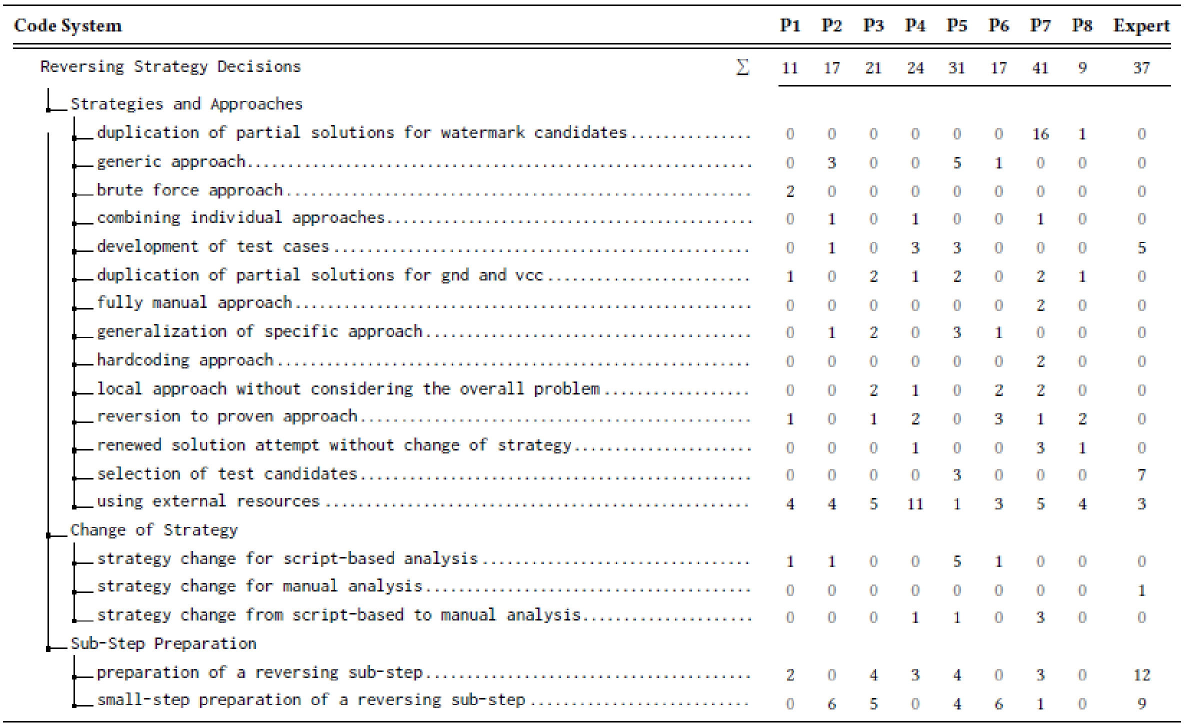
Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Test and Validation	∑	27	12	30	32	45	14	22	7	43
Focused										
manual netlist inspection for script validation		3	1	3	16	7	0	3	1	4
targeted evaluation		7	3	12	6	12	4	7	5	13
targeted verification		8	5	7	6	13	1	3	1	14
testing python constructs		3	1	0	1	1	2	1	0	6
testing the hal api		0	0	1	0	0	0	0	0	
General										
general evaluation		3	2	5	2	12	0	7	0	3
reliability test		3	0	2	1	0	7	1	0	3

Table 11. Hierarchy of the sub-category Code Adjustments and absolute number of assigned open codes per participant.



Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Code Adjustments	∑	49	39	42	49	70	38	41	20	63
Creating Clarity										
cleaning up within the code		11	6	9	10	4	2	4	3	12
code optimization for error avoidance		1	1	1	0	7	2	1	0	4
code restructuring		0	2	0	0	6	2	1	0	4
code simplification		0	3	3	1	0	4	5	0	4
improve clarity of console output		17	7	11	8	23	11	5	8	19
improve clarity of code		2	6	7	6	5	7	5	1	7
Cut, Copy, and Paste										
code artifact remains		0	1	0	1	0	0	0	0	0
deletion of actually needed code		2	0	0	1	0	0	0	0	0
introduction of redundant code		0	0	1	1	0	0	3	1	0
reversion to previous code components		3	3	3	9	6	3	7	0	4
restoration of a functional code state		1	0	4	2	5	0	3	2	1
Documentation										
anticipatory documentation		0	2	1	0	2	0	3	0	0
documentation of results		9	6	1	7	0	4	2	3	5
explanatory documentation		3	2	1	3	12	3	2	2	3

Table 12. Hierarchy of the sub-category Reversing Strategy Decisions and absolute number of assigned open codes per participant.

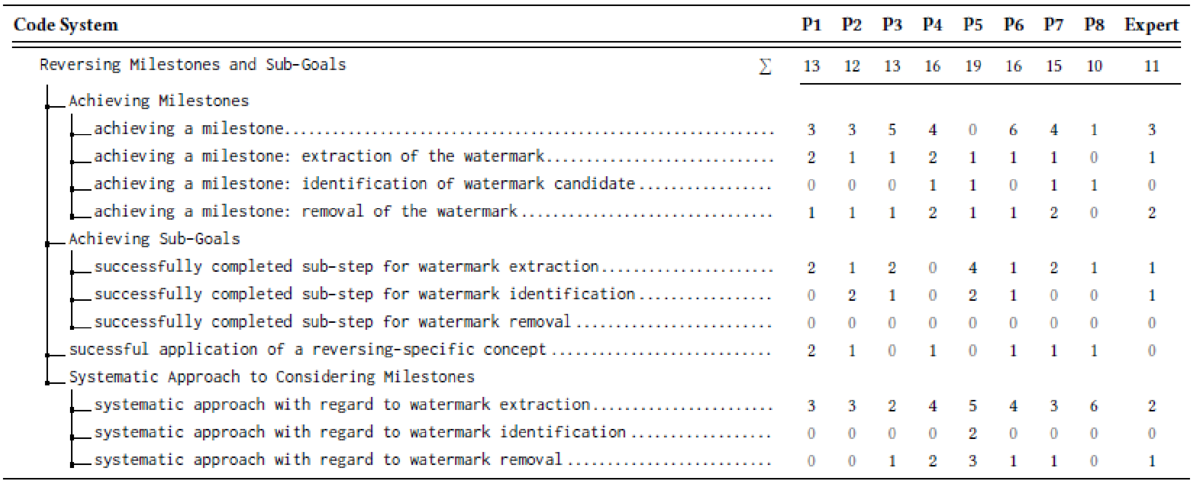


Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Reversing Strategy Decisions	Σ	11	17	21	24	31	17	41	9	37
Strategies and Approaches										
duplication of partial solutions for watermark candidates		0	0	0	0	0	0	16	1	0
generic approach		0	3	0	0	5	1	0	0	0
brute force approach		2	0	0	0	0	0	0	0	0
combining individual approaches		0	1	0	1	0	0	1	0	0
development of test cases		0	1	0	3	3	0	0	0	5
duplication of partial solutions for gnd and vcc		1	0	2	1	2	0	2	1	0
fully manual approach		0	0	0	0	0	0	2	0	0
generalization of specific approach		0	1	2	0	3	1	0	0	0
hardcoding approach		0	0	0	0	0	0	2	0	0
local approach without considering the overall problem		0	0	2	1	0	2	2	0	0
reversion to proven approach		1	0	1	2	0	3	1	2	0
renewed solution attempt without change of strategy		0	0	0	1	0	0	3	1	0
selection of test candidates		0	0	0	0	3	0	0	0	7
using external resources		4	4	5	11	1	3	5	4	3
Change of Strategy										
strategy change for script-based analysis		1	1	0	0	5	1	0	0	0
strategy change for manual analysis		0	0	0	0	0	0	0	0	1
strategy change from script-based to manual analysis		0	0	0	1	1	0	3	0	0
Sub-Step Preparation										
preparation of a reversing sub-step		2	0	4	3	4	0	3	0	12
small-step preparation of a reversing sub-step		0	6	5	0	4	6	1	0	9

Table 13. Hierarchy of the sub-category Inspection and Information Gathering and absolute number of assigned open codes per participant.

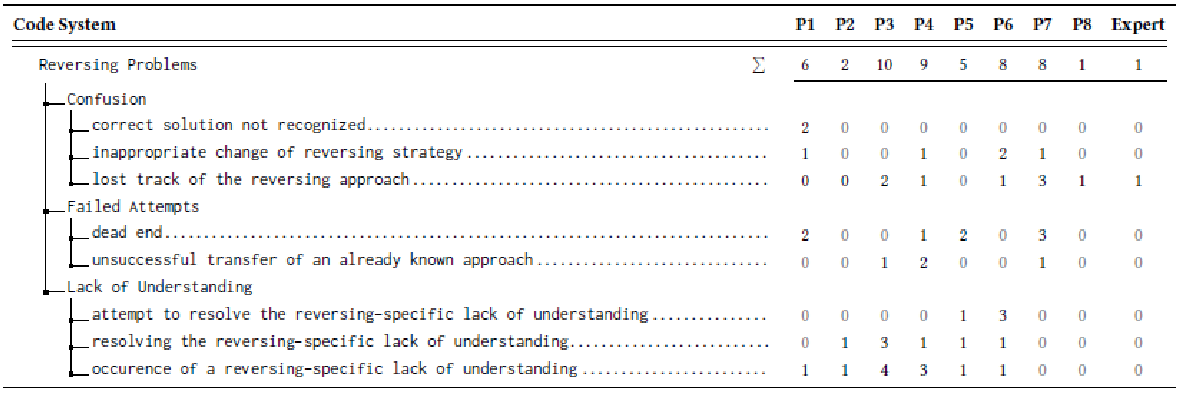
Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Inspection and Information Gathering	Σ	10	17	15	11	23	17	25	19	23
Manual Actions										
Manual Exploration										
manual netlist exploration		1	2	1	0	2	1	1	4	1
manual selection of subsequent gate from gnd or vcc		0	1	0	0	0	1	0	0	0
manual selection of irrelevant gates		1	1	2	0	0	1	2	0	7
manual selection of irrelevant nets		0	1	1	0	0	1	0	2	0
manual selection of watermark candidates		0	0	0	0	2	0	0	0	5
Manual Identification										
first manual identification of the gnd gate		0	1	0	0	0	0	1	1	0
first manual identification of the gnd net		1	0	0	1	1	1	1	1	0
first manual identification of the vcc gate		0	0	0	0	1	0	0	1	0
first manual identification of the vcc net		1	1	0	1	1	1	0	1	0
Manual Inspection										
in-depth manual inspection of watermark candidates		0	2	0	5	7	3	17	3	2
manual inspection of irrelevant gates		4	1	0	1	2	0	0	1	1
manual inspection of the gnd gate		0	1	1	1	0	1	2	1	0
manual inspection of the gnd net		1	0	0	1	1	2	1	2	0
manual inspection of the vcc gate		0	0	0	0	5	1	0	0	0
manual inspection of the vcc net		1	1	0	1	1	1	0	2	0
Script-based Actions										
Script-based Exploration										
script-based netlist exploration		0	1	4	0	0	1	0	0	2
script-based search for gnd and vcc nets		0	0	2	0	0	0	0	0	0
Script-based Identification										
script-based identification of the gnd gate		0	0	1	0	0	0	0	0	1
script-based identification of the vcc gate		0	0	0	0	0	0	0	0	1
Script-based Inspection										
script-based netlist inspection		0	0	0	0	0	0	0	0	1
script-based inspection of global inout nets		0	0	1	0	0	0	0	0	0
script-based inspection of input nets		0	3	0	0	0	0	0	0	0
script-based inspection of the gnd and vcc nets		0	0	0	0	0	2	0	0	0
script-based inspection of the input pin types		0	1	0	0	0	0	0	0	0
script-based inspection of watermark candidates		0	0	2	0	0	0	0	0	2

Table 14. Hierarchy of the sub-category Reversing Milestones and Sub-Goals and absolute number of assigned open codes per participant.



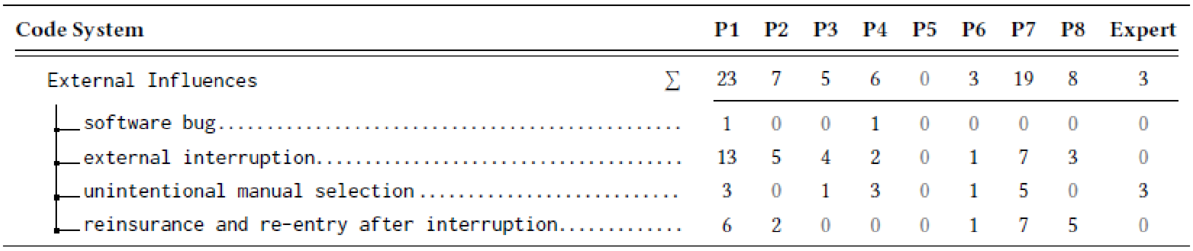
Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Reversing Milestones and Sub-Goals	∑	13	12	13	16	19	16	15	10	11
Achieving Milestones										
achieving a milestone		3	3	5	4	0	6	4	1	3
achieving a milestone: extraction of the watermark		2	1	1	2	1	1	1	0	1
achieving a milestone: identification of watermark candidate		0	0	0	1	1	0	1	1	0
achieving a milestone: removal of the watermark		1	1	1	2	1	1	2	0	2
Achieving Sub-Goals										
successfully completed sub-step for watermark extraction		2	1	2	0	4	1	2	1	1
successfully completed sub-step for watermark identification		0	2	1	0	2	1	0	0	1
successfully completed sub-step for watermark removal		0	0	0	0	0	0	0	0	0
sucessful application of a reversing-specific concept		2	1	0	1	0	1	1	1	0
Systematic Approach to Considering Milestones										
systematic approach with regard to watermark extraction		3	3	2	4	5	4	3	6	2
systematic approach with regard to watermark identification		0	0	0	0	2	0	0	0	0
systematic approach with regard to watermark removal		0	0	1	2	3	1	1	0	1

Table 15. Hierarchy of the sub-category Reversing Problems and absolute number of assigned open codes per participant.



Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
Reversing Problems	∑	6	2	10	9	5	8	8	1	1
Confusion										
correct solution not recognized		2	0	0	0	0	0	0	0	0
inappropriate change of reversing strategy		1	0	0	1	0	2	1	0	0
lost track of the reversing approach		0	0	2	1	0	1	3	1	1
Failed Attempts										
dead end		2	0	0	1	2	0	3	0	0
unsuccessful transfer of an already known approach		0	0	1	2	0	0	1	0	0
Lack of Understanding										
attempt to resolve the reversing-specific lack of understanding		0	0	0	0	1	3	0	0	0
resolving the reversing-specific lack of understanding		0	1	3	1	1	1	0	0	0
occurence of a reversing-specific lack of understanding		1	1	4	3	1	1	0	0	0

Table 16. Hierarchy of the sub-category External Influences and absolute number of assigned open codes per participant.



Code System		P1	P2	P3	P4	P5	P6	P7	P8	Expert
External Influences	∑	23	7	5	6	0	3	19	8	3
software bug		1	0	0	1	0	0	0	0	0
external interruption		13	5	4	2	0	1	7	3	0
unintentional manual selection		3	0	1	3	0	1	5	0	3
reinsurance and re-entry after interruption		6	2	0	0	0	1	7	5	0

9.5 List of Abbreviations

ACT-R	Adaptive Control of Thought-Rational
AES	Advanced Encryption Standard
CPS	Complex problem solving
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GND	Ground
GUI	Graphical User Interface
HCI	Human Computer Interaction
HRE	Hardware Reverse Engineering
IC	Integrated circuit
IoT	Internet of Things
IP	Intellectual property
IRB	Institutional review board
LTM	Long-term-memory
LUT	Look-up Table
PR	Perceptual Reasoning
PS	Processing Speed
QCM	Questionnaire on Current Motivation
RAT	Remote Associates Test
RQ	Research question
SRE	Software Reverse Engineering
VCC	Voltage common collector
WAIS	Wechsler Adult Intelligence Scale
WM	Working Memory

9.6 List of Figures

9.7 List of Tables

9.8 About the Author

Curriculum Vitae

Personal Data

Name: Carina Yasmin Wiesen

Address:
Ruhr-Universität Bochum
Universitätsstr. 150
D – 44780 Bochum

Education

since 2017	**Ph.D. Student**, Ruhr-Universität Bochum, Germany
2015	**M.Sc.**, Applied Cognitive Science and Media Science, Universität Duisburg-Essen
2013	**B.Sc.**, Applied Cognitive Science and Media Science, Universität Duisburg-Essen
2007	**Abitur**, Marienschule Opladen, Gymnasium des Erzbistums Köln, Opladen, Germany.

Peer-Reviewed Publications in Journals

Wiesen, C., Becker, S., Walendy, R.; Paar, C., & Rummel, N. (under review). *The Anatomy of a Hardware Reverse Engineering Attack: Insights into Cognitive Processes during Problem Solving.* ACM Transactions on Computer-Human Interaction TOCHI.

Peer-Reviewed Publications in Conferences / Workshops

Becker, S., Wiesen, C., Albartus, N., Rummel, N., & Paar, C. (2020). An Exploratory Study of Hardware Reverse Engineering – Technical and Cognitive Processes. Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020).

Wiesen, C., Becker, S., Paar, C., & Rummel, N. (2019). Promoting Skill Acquisition in Hardware Reverse Engineering. In Proceedings of the *2019 IEEE Frontiers in Education Conference (FIE)*, Cincinnati, OH, USA, 2019.

Wiesen, C., Becker, S., Fyrbiak, M., Albartus, N., Elson, M., Rummel, N., & Paar, C. (2018). Teaching Hardware Reverse Engineering: Educational Guidelines and Practical Insights. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 438-445). IEEE.

Participation in Selected Conferences and Workshops

Becker, S., & Wiesen, C. (2020). Towards Cognitive Obfuscation. RuhrSec IT Security Conference, Ruhr University Bochum, May, 2020.

Wiesen, C. (2019). *Towards Cognitive Obfuscation*: *Analyzing Human Factors to Impede Hardware Reverse Engineering*. Talk at the International Workshop on Cryptography, Robustness, and Provably Secure Schemes for Female Young Researchers (CrossFyre) at TU Darmstadt (co-located to Eurocrypt 2019), May 2019.

Wiesen, C., Becker, S., Paar, C., & Rummel, N. (2019). *Acquisition of Hardware Reverse Engineering Competency in IT Security – An Explorative Field Study*. Paper presented at the European Association for Research on Learning and Instruction (EARLI) in Aachen, Germany, August 2019.

Becker, S., Wiesen, C., Albartus, N., Wallat, S., Rummel, N., & Paar, C. (2019). Poster presented at IACR Transactions on Cryptographic Hardware and Embedded Systems, CHES 2019, Atlanta, USA, August 26 – 28, 2019)

Wiesen, C., Elson, M., Fyrbiak, M., Becker, S., Paar, C., & Rummel, N. (2018). *Hardware Reverse Engineering als eine spezielle Art des Problemlösens.* Vortrag auf dem 51. Kongress der Deutschen Gesellschaft für Psychologie (DGPs), 15.09.-20.09.2018, Frankfurt am Main.

Becker, S., Wiesen, C., Fyrbiak, M., Rummel, N., & Paar, C. (2018). *Hardware Reverse Engineering & Cognitive Countermeasures.* Poster and Demo presented at the Intel-CRI-CARS Workshop 2018 at Intel, Hillsboro Oregon (17-18 May 2018).

Invited Publications (Not peer-reviewed)

Wiesen, C., Albartus, N., Hoffmann, M., Becker, S., Wallat, S., Fyrbiak, M., Rummel, N., & Paar, C. (2019). Towards cognitive obfuscation: impeding hardware reverse engineering based on psychological insights. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (pp. 104-111). ACM.

Becker, S., Wiesen, C., Paar, C., & Rummel, N. (2019). Wie arbeiten Reverse Engineers?. *Datenschutz und Datensicherheit-DuD, 43*(11), 686-690.

9.9 Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die eingereichte Dissertation selbstständig und ohne unzulässige fremde Hilfe verfasst, andere als die in ihr angegebene Literatur nicht benutzt und dass ich alle ganz oder annähernd übernommenen Textstellen sowie verwendete Grafiken, Tabellen und Auswertungsprogramme kenntlich gemacht habe. Außerdem versichere ich, dass die vorgelegte elektronische mit der schriftlichen Version der Dissertation übereinstimmt und die Abhandlung in dieser oder ähnlicher Form noch nicht anderweitig als Promotionsleistung vorgelegt und bewertet wurde.

Weiterhin erkläre ich, dass digitale Abbildungen nur die originalen Daten oder eine eindeutige Dokumentation von Art und Umfang der inhaltsveränderten Bildbearbeitung enthalten.

Ich versichere ebenfalls, dass keine kommerzielle Vermittlung oder Beratung in Anspruch genommen wurde.

____

Unterschrift