An Exploratory Study of Hardware Reverse Engineering — Technical and Cognitive Processes

Steffen Becker, Carina Wiesen, and Nils Albartus, *Ruhr University Bochum, Max Planck Institute for Cybersecurity and Privacy;* Nikol Rummel, *Ruhr University Bochum;* Christof Paar, *Max Planck Institute for Cybersecurity and Privacy*

https://www.usenix.org/conference/soups2020/presentation/becker

This paper is included in the Proceedings of the Sixteenth Symposium on Usable Privacy and Security.

August 10–11, 2020

978-1-939133-16-8

Open access to the Proceedings of the Sixteenth Symposium on Usable Privacy and Security is sponsored by USENIX.

An Exploratory Study of Hardware Reverse Engineering Technical and Cognitive Processes

Steffen Becker*†‡, Carina Wiesen*†‡, Nils Albartus†‡, Nikol Rummel†, and Christof Paar‡
†*Ruhr University Bochum;* ‡*Max Planck Institute for Cybersecurity and Privacy*
{steffen.becker, carina.wiesen, nils.albartus, nikol.rummel}@rub.de; christof.paar@csp.mpg.de
**Both authors contributed equally to this work.*

Abstract

Understanding the internals of Integrated Circuits (ICs), referred to as Hardware Reverse Engineering (HRE), is of interest to both legitimate and malicious parties. HRE is a complex process in which semi-automated steps are interwoven with human sense-making processes. Currently, little is known about the technical and cognitive processes which determine the success of HRE.

This paper performs an initial investigation on how reverse engineers solve problems, how manual and automated analysis methods interact, and which cognitive factors play a role. We present the results of an exploratory behavioral study with eight participants that was conducted after they had completed a 14-week training. We explored the validity of our findings by comparing them with the behavior (strategies applied and solution time) of an HRE expert. The participants were observed while solving a realistic HRE task. We tested cognitive abilities of our participants and collected large sets of behavioral data from log files. By comparing the least and most efficient reverse engineers, we were able to observe successful strategies. Moreover, our analyses suggest a phase model for reverse engineering, consisting of three phases. Our descriptive results further indicate that the cognitive factor Working Memory (WM) might play a role in efficiently solving HRE problems. Our exploratory study builds the foundation for future research in this topic and outlines ideas for designing cognitively difficult countermeasures ("cognitive obfuscation") against HRE.

USENIX Symposium on Usable Privacy and Security (SOUPS) 2020.
August 9–11, 2020, Virtual Conference.

1 Introduction

By definition, every computing system is based on hardware components, in particular on Integrated Circuits (ICs). Their internals are typically completely opaque to the user and largely even to the developers of the system. Understanding the internals of (digital) hardware components, which requires Hardware Reverse Engineering (HRE), is of interest for both malicious and legitimate reasons [27]. For instance, the sensitive topic of low-level backdoors (i.e., hardware Trojans), which underlies the current discussion about foreign-built communication and computer equipment [29, 30], requires HRE for detection of such manipulations. On the other hand, adversaries might also need to reverse engineer the hardware they plan to subvert. HRE is also widely-used in practice for detection of Intellectual Property (IP) infringements [13]. On the adversary side, a malicious party needs to reverse areas of interest or even entire ICs. Moreover, there is a host of Trojan detection techniques [33] that require a flawless model of the target IC.

Despite its relevance, HRE is relatively poorly understood compared to many other areas of both hardware design and computer security [13]. We argue that it is desirable to obtain a better understanding of HRE. First, it will aid with assessing the threat posed by adversaries performing HRE. This is particularly prudent because there is undoubtedly expertise about HRE within government agencies with malicious intent [20]. Second, HRE performed for defensive purposes such as IC verification will benefit from having a better understanding of the involved time and costs. Third, having a better grasp on HRE will allow us to derive sound obfuscation techniques that exploit cognitive limitations of humans.

HRE is a multilayered process, where high-level information is extracted from a low-level circuit, consisting of two major stages [1]: In the first stage, a gate-level netlist is obtained from an IC either directly from the device or possibly through (online) interception of design information. A gate-level netlist is a logical circuit description and is usually composed of Boolean gates and their respective interconnections.

The technical steps required for netlist extraction, including decapsulation and imaging, are relatively well-covered in the literature [27, 35]. The objective of the second stage is to *make sense* of the recovered netlist, that is, to *understand* the netlist [1]. This second stage has only been rudimentarily addressed in the literature and is the topic of our investigation.

In this contribution, we present an exploratory study with the goal of obtaining initial insights into the complex processes of the sense-making part of HRE. This undertaking is particularly challenging as it depends on non-trivial technical steps as well as on cognitive factors of the human reverse engineer. To the best of our knowledge, it is the first time such a study based on observing a group of reverse engineers has been conducted.

Overview on our Exploratory Study

In order to gain such insights, we conducted an empirical study that explores both the technical and the underlying cognitive processes of hardware reverse engineers. In an ideal scenario, we would examine expert reverse engineers working on real-world tasks with tools they are familiar with. However, we face the methodological problem that those experts are few to begin with. They are primarily active in government agencies and a few highly specialized companies, and are generally unavailable to the scientific community. We approach this problem by observing students enrolled in a 5-years BSc-MSc program in cyber security while solving a realistic HRE task, which was developed in collaboration with HRE experts. Prior to the study, the students had been exposed to an extensive 14-week HRE training. For the study, we selected eight students based on their performance during the HRE training.

Despite the difficulty of engaging HRE experts as study participants, we were able to recruit one expert via the professional network of one of the authors. This expert solved the same task as our participants. The collected data served as a sanity check for our student sample with respect to solution time, as well as phases of and applied strategies during HRE.

In our exploratory study, we collected detailed behavioral data during an HRE-based attack and measured cognitive factors of eight participants. The HRE task is based on a realistic setting, where the analyst has to circumvent an IP protection mechanism in an unknown circuit. Analysis of the data revealed initial insights into problem solving strategies and relevant cognitive factors in HRE. We were also able to derive first hypotheses for a novel class of obfuscation measures that take the boundaries of cognitive abilities into account. Although our work was faced with methodological challenges that we discuss in the limitations section, we render the following main contributions advancing the current scientific knowledge of technical and cognitive processes in HRE.

1) We propose and examine an HRE phase model based on an exploratory behavioral study of human reverse engineers, who solved a realistic HRE task.
2) Based on behavioral analyses we explore more and less efficient HRE strategies of reverse engineers.
3) We explore the role of different cognitive factors in efficiently solving HRE problems.
4) Based on our findings, we derive hypotheses for a novel class of HRE countermeasures called cognitive obfuscation and outline future research directions.

2 Background

In this section, we present the relevant technical background of HRE and propose a three-phase model encompassing human processes during HRE. Furthermore, we introduce related work on cognitive processes in reverse engineering. Against this background, we identify the research gap and derive research questions we seek to answer in this work.

2.1 Hardware Reverse Engineering

Reverse engineering is the process of extracting knowledge or design information from anything man-made in order to comprehend its inner structure [28]. As mentioned above, in the case of HRE, there are two distinct stages [1]. In the first stage, a gate-level netlist is obtained directly from an IC or a Field Programmable Gate Array (FPGA) or through (online) interception of design information. Although netlist extraction requires sophisticated technical methods, research has shown that netlists can be extracted reliably by trained specialists from both, ICs and FPGAs [9, 23, 35].

In the second, sense-making stage of HRE, the netlist is transformed into higher levels of abstraction that enable a detailed analysis. This often involves module recognition, identification of blocks of interest, and detailed understanding of Boolean sub-circuits [1, 14, 32]. The analysis typically serves a specific objective, for example, finding and understanding IP blocks or extraction of cryptographic keys [39]. Due to the nature of this stage — which requires human ingenuity, sense-making, and in many cases customized solutions — fully automated tools do not exist [13]. Instead, the analyst typically employs HRE tools which enable interaction with the target netlist. Tools may provide semi-automated support for the human analyst, for example, for running specific algorithms on the netlist [8], as well as, features for manual analysis of netlist components [38].

Even with tool support, the cognitive processes and human problem-solving strategies are crucial for HRE, yet remain poorly understood. Against this background, it is hardly surprising that hardware obfuscation, which is a widely used countermeasure against HRE, is largely based on ad-hoc methods [41]. We argue that a comprehension of human processes in HRE will open a pathway for the development of novel obfuscation techniques. This paper will conclude with first guidelines for how such cognitive obfuscation measures might look like.

Reverse engineering of large and complex netlists is commonly driven by an objective more narrow than full understanding of the entire netlist. We model a situation under which HRE is typically performed in practice through the following conditions:

- An (error-free) netlist of the entire design or the area of interest is at hand.
- An HRE tool is available that allows interaction with the target netlist.
- There is a clear objective, for example, removal of an IP protection mechanism.
- (First) hypotheses related to the objective exist, for example, which IP protection mechanism is implemented.

If these preconditions are met, a human analyst attempts to understand the target netlist through two principal means:

- **Manual analyses**: Detailed manual inspection of netlist components and explorative navigation through textual and graphical representations of the netlist.
- **Semi-automated analyses**: Customized scripts and programs that facilitate structural and functional analyses of the netlist.

In Section 4, we will explore which role these two mechanisms play during HRE.

2.2 Phase Model for Gate-level Netlist Reverse Engineering

Due to the sheer complexity of the sense-making stage, it is plausible that human strategies during this process can be divided into sub phases of human sense-making. Thus, we propose a three-phase model derived from several HRE-based attacks from literature [1, 14], and two technical HRE workflow descriptions [8, 32]. Even though not explicitly formulated in a model before, the three phases are an implicit hypothesis about the inner workings of HRE. In a very recent work, Votipka et al. introduce a similar model of human processes during software reverse engineering based on expert interviews [37]. In the following passages, we briefly characterize each phase under the assumption that the preconditions described above are fulfilled. Also, since netlist reverse engineering is a continuous process, the consecutive phases can blend into each other rather than being strictly disjoint.

Phase 1: Candidate Identification

The goal of Phase 1 is to identify single candidates and sub-circuits which are potentially relevant in the context of the reverse engineering process. Therefore, the analyst starts exploration via structural analyses of the netlist topology with the goal to identify blocks of interests. The result are sub-circuit candidates, which are further inspected in Phase 2. Suited methods for this phase are semi-automated structural analyses (e.g., graph clustering algorithms or sub-circuit matching) as well as custom-tailored structural analyses incorporating hypotheses about searched structures. Additionally, manual netlist exploration can provide a starting point for the reverse engineer, for example, by inspecting global in- and outputs.

Phase 2: Candidate Verification

The goal of Phase 2 is the verification of extracted candidates from Phase 1 in order to narrow them down and select target components for Phase 3. If no target components are remaining, new candidates have to be identified in Phase 1 iteratively by refining the methods used. Phase 2 incorporates static analyses methods such as Boolean functionality analysis or sub-circuit matching. Manual inspection of candidates can support the reverse engineer by testing the existing hypotheses before solving the problem algorithmically.

Phase 3: Realization

While Phases 1 and 2 can be generalized for most HRE tasks, the goals and procedures of Phase 3 differ significantly depending on the objective. Methods employed include sub-circuit interpretation and annotation in order to obtain a more abstract netlist model; netlist simulation to analyze sequential behavior; or preparation of malicious netlist manipulation, for example, add, remove, or change functionality of netlist components. In many cases, Phase 3 incorporates and combines several of the aforementioned methods.

2.3 Cognitive Processes in Reverse Engineering

As outlined above, HRE always involves sense-making processes. Thus, the success of HRE heavily depends on skills, knowledge, and expertise of the performing reverse engineer. Surprisingly, underlying cognitive processes in HRE are understudied and remain poorly understood [13]. Despite this observation, prior research on HRE almost solely focuses on technical factors. Nonetheless, one prior work explored cognitive processes by defining reverse engineering of Boolean systems as a specific type of human problem solving [19]. In general, a problem exists when a person lacks in knowledge which enables the problem solver to achieve a desired goal [12]. Problem solving is defined as a sequence of cognitive operations (e.g., problem solving strategies) in order to solve a task for which the individual does not possess a suitable routine operation [25]. The success of problem solving is influenced by several factors, for example, prior domain specific knowledge [5], or cognitive abilities (e.g., intelligence and sub factors like Working Memory) [17]. Baddeley demonstrated that brain systems like the Working Memory are essential in solving complex cognitive tasks like problem solving [2]. Besides cognitive abilities, the level of expertise [7] is another important factor in determining problem solving performances. Larkin et al. showed that experts were quicker in solving physics problems than novices [18].

In 2013, Lee and Johnson-Laird analyzed problem solving behavior in reverse engineering of Boolean systems by

conducting five experiments in a laboratory setting [19]. The tasks students were asked to solve merely involved drawing Boolean circuits controlling an electric light. Consequently, the ecological and external validity of these experiments appears low in the context of HRE. Thus, it remains unclear to what extent the results of Lee and Johnson-Laird can be generalized to reverse engineering of entire ICs, which commonly consist of hundred of thousands or millions of logic components. Nevertheless, we transfer human problem solving processes to the cognitive processes in HRE by observing problem solving strategies of more and less successful reverse engineers and by measuring cognitive factors which might play a role in HRE problem solving.

2.4 Research Gap and Research Questions

In this work, we aim to close the existing research gap by providing first insights into the technical and cognitive processes during a realistic HRE task. We qualitatively examine the occurrence of the 3-phase model for netlist reverse engineering in the participants' behavior, and investigate HRE problem solving strategies and their influencing cognitive factors on the basis of behavioral and cognitive data. This allows us to derive hypotheses for cognitive obfuscation techniques, i.e., novel countermeasures impeding HRE. The paper at hand attempts to answer the following research questions:

RQ1. Can the phases of human sense-making be detected during HRE processes? If so, which are the crucial phases?

RQ2. Which strategies distinguish more and less efficient reverse engineers?

RQ3. Which cognitive prerequisites play a role for the success of HRE?

RQ4. How can those insights be used to derive hypotheses for cognitive obfuscation?

Our exploratory study opens many venues for further in-depth research on the challenging interdisciplinary problem of understanding HRE.

3 Methodology

In the following section, we first describe the research environment enabling our study. Second, we outline the details of our user study, including our participants and all study related measures and processes. Last, we explain the behavioral analyses providing the underlying data for our exploration of human factors in HRE.

3.1 Research Environment

This section outlines the research environment consisting of an extensive HRE training, the HRE tool HAL, and a practical HRE task under study.

3.1.1 HRE Training

The contents of the 14-week HRE training were developed based on the comprehensive body of technical research and industry practices and on educational guidelines facilitating learning HRE with input from experts from academia and practice [43]. Subsequently, it was shown that the training successfully promotes HRE skill acquisition [42]. During the first six weeks, the instructors conveyed necessary theoretical backgrounds, before students solved four training tasks with the HRE tool HAL in the 8-week practical part. The training was followed by a two-week study, where participants solved a realistic HRE task. Crucially, neither the proposed HRE phase model, nor the concrete scenario of, or any solution strategies for the Study Task were part of the training.

3.1.2 HRE Tool HAL

Given today's integration density, which commonly results in very complex netlists, it is virtually impossible to reverse even moderately sized IC or FPGA designs without tool support. Therefore, it is safe to assume that professional reverse engineers, for example, in government agencies and specialized companies, have access to such (internally developed) tools, cf. [20, 34]. Recently, the open-source framework HAL [15, 38] has become available on GitHub, which is the first tool specifically designed to facilitate HRE. By using HAL in our study, we create an environment that is, thus, similar to one encountered in real-world HRE situations.

HAL operates on gate-level netlists. It has a modular and extendable design and supports both, static analysis and cycle-accurate simulation of netlists. There are several references describing semi-automated reverse engineering and manipulation tasks using HAL [14, 15, 39]. Crucially for our study, HAL can capture all user interactions and therefore enables the investigation of the many sub-steps that take place during HRE, including the study of human factors. Below is a description of HAL features that were particularly useful for our study:

- HAL offers an *interactive GUI* allowing detailed manual inspection and exploration of netlist components as well as module grouping functionalities.
- HAL natively implements a *Python interface* enabling script-based interactions with the netlist.
- HAL comes with a variety of *accompanying materials*, most importantly a detailed documentation of HAL-specific Python commands and a coding guide providing many examples for common use cases.

3.1.3 Tasks and Materials

In the practical part of the HRE training and in the study, participants worked on five reverse engineering tasks based on flat netlists synthesized for FPGAs. Those netlists do not contain any high-level information about the design such as component names, module boundaries or hierarchy elements. Netlist components are composed of Look-up tables (LUTs)

with up to six inputs, which realize the circuit's Boolean functions, multiplexers, Flip Flops (FFs), and their respective logic interconnections.

All five tasks are based on ideas drawn from recent literature on HRE attacks and countermeasures [1, 6, 14]. Thus, they represent a number of different HRE settings ranging from sub-circuit detection over obfuscation circumvention up to malicious manipulation of cryptographic cores. The tasks are designed with increasing difficulty, that is, the problem itself increased in complexity, the level of guidance decreased, and the size and complexity of the target netlist increased, cf. Table 1. The level of guidance includes task-related support by instructors, which participants received only during the training tasks, as well as the amounts of accompanying materials, e.g., (excerpts of) scientific papers, or relevant examples from the coding guide for the task.

Table 1: Difficulty, level of guidance, and netlist complexity for the HRE tasks on a scale from **+** (low) to **+++** (high).

	Difficulty	**Guidance**	**Complexity**
Training Task 1	+	+++	+
Training Task 2	++	+++	+
Training Task 3	++	++	+
Training Task 4	+++	++	++
Study Task	+++	+	++

In the following, the last and most difficult task, which provides the data for our study, is introduced.

Study Task: Breaking Watermarkings. In this task, the IP protection mechanism—a so-called watermarking scheme—proposed by Schmid et al. [31] is embedded in a hardware design, enabling the detection of IP theft. The analyst has the objective to clone the underlying netlist without copying its watermark, and therefore thwarting detection of IP theft in the cloned circuit. While materials such as the original paper on the functionality of the watermarking scheme were provided, the analysts had to develop the strategies for detection and removal of the watermarking themselves.

To enable analysis of HRE strategies employed by the participants while solving the Study Task, netlist components are pre-annotated in relevant and irrelevant ones. Out of 4653 total netlist components, 54 were marked as relevant since they are associated with the watermark. 50 of those components are *candidates*, thereof 30 actually implement the watermark and are therefore *targets*. The remaining 4 components are important anchor points for the watermark detection. Naturally, the annotation was invisible to the participants. All netlist components have a unique identifier in HAL, which allows their tracing during the behavioral analyses described in Section 3.3.

3.2 User Study and Variables

This section describes our user study in detail. This includes the specification of our study participants and the expert, ethical considerations concerning this research, as well as an outline of the study procedures and variables.

3.2.1 Study Participants

We conducted our quasi-experimental study with a within-subject design, where every participant had to solve the Study Task using HAL. Since experts were generally unavailable for this research, we approximated experienced reverse engineers by recruiting senior-level and MSc-level students with a (target) degree in cyber security. The recruited participants acquired relevant HRE knowledge and skills during the extensive training phase and were selected based on their performance in the four training tasks described in Section 3.1.1. Furthermore, we approximated the situation of experienced reverse engineers by providing a coding guide containing relevant code snippets, instructions, and programming methods which were used throughout the study and the training tasks.

Overall, 22 participants volunteered to participate in the study. All participants were enrolled in either the last year of a three-year bachelor's or in a master's program in cyber security. Five of the original 22 participants decided to drop out. Furthermore, three participants were excluded due to incomplete datasets. Out of the remaining 14, eight participants (mean age: 24 years; $SD = 4$ years; one graduate student) were selected based on their performance in the training tasks, i.e., the average solution probability of all training tasks was above 95% per selected participant. Solution probabilities were evaluated on a scale from 0% to 100% by three teaching assistants based on a detailed gradebook with sample solutions.

3.2.2 Sanity Check with Expert

Despite the difficulty of recruiting HRE experts as study participants, we were able to recruit one expert through the professional network of one of the authors. The expert also performed the Study Task. There was a two-fold objective to engaging the expert. First, the comparison of the reversing behavior between the expert and the study participants allowed us to explore if the students' approximated level as experienced reverse engineers was an adequate assumption. Second, we could evaluate the difficulty of the Study Task. For assessing the status as HRE expert, we followed the criteria of Votipka et al. [37]: Our expert had 5 years of experience and a self-assessed skill-level of 4 on a 5-point Likert-Scale (with 1 being a beginner and 5 being an expert). The expert gave written informed consent on using the collected socio-demographic and behavioral data in the context of this research project. Cognitive abilities were not tested for the expert to protect the person's privacy in case of a potential de-anonymization.

The expert received the underlying netlist of the Study Task and corresponding materials (virtual machine running HAL, coding guide, documentation of Python commands, and paper on the implemented watermarking) and was asked to remove the IP protection mechanism from the circuit. Moreover, the expert was already experienced in working with HAL. Subsequently, the behavioral data collected while the expert was working on the Study Task was analyzed in order to compare the observations made for our student sample with respect to solution time, as well as phases of and applied strategies during HRE with the expert's problem solving behavior.

3.2.3 Control Variables

We asked participants to provide information about their socio-demographics (age, major, target degree, number of semesters enrolled) in a self-developed questionnaire (cf. Appendix A).

3.2.4 Cognitive Abilities

Based on the definition of reverse engineering as a specific type of problem solving [19], we transferred the concept of problem solving research into the domain of HRE. Therefore, as problem solving performances can depend on cognitive abilities [17], we measured sub factors of general intelligence and their correlations to HRE problem solving performances. As a measure for problem solving performance, we correlate the variable time on task, which is a traditional measure in cognitive psychology, with levels of cognitive abilities. In order to assess the participants' levels of cognitive abilities, sub tests of a valid test instrument, the Wechsler Adult Intelligence Scale (WAIS-IV) [40], were used. We integrated the following three sub scores in our study: Perceptual Reasoning (PR), Working Memory (WM), and Processing Speed (PS). The fourth test of the WAIS-IV on Verbal Comprehension (VC), which measures verbal reasoning and verbal expression, was not included in the study since participants had different native languages. PR measures the ability to accurately interpret and work with visual information. The sub score WM assessed the ability to store information and to perform mental operations on that stored information. The third score PS reflected the ability of processing visual information quickly and efficiently.

3.2.5 Ethical Considerations

Our institute does not have an ethics board or IRB, but the study protocols were reviewed and approved by the universities' data protection officer. Before entering the HRE training, all 22 participants gave written informed consent. They received monetary compensation well above minimum wage levels for time spent on materials related to our study. We informed participants that they can withdraw from our study at any time and that all partial data will not be analyzed or stored. Privacy was ensured by randomly assigning pseudonyms to the participants, which were used instead of their actual names throughout all materials related to our study.

3.2.6 Study Procedures

The WAIS-IV was conducted in a 60 to 90 minutes face-to-face session with each participant prior to the study. Before starting the Study Task, participants were asked to answer the questionnaire on socio-demographics via the online survey provider Soscisurvey. After finishing the Study Task, participants uploaded log files recorded while solving the Study Task onto a server located at the university.

3.3 Behavioral Analyses

In order to explore human processes in HRE, we analyzed log files automatically generated by HAL. We collected one log file — containing several thousands up to tens of thousands of log entries — for each participant solving the Study Task. Every log entry consists of a timestamp and one of the following events in HAL: (i) content and terminal output of the executed Python script; (ii) manual selection of netlist component (unique identifier) via GUI; or (iii) additional system-level entries such as indications for user (in)activity. The following parameters were extracted from the collected logfiles and serve as the basis for the behavioral analyses.

Total Solution Time. Since participants solved the task over a period of two weeks, relative solution time was calculated, where loading the inspected netlist for the first time in HAL marks the starting time. All periods lasting more than 10 minutes without log entries were manually reviewed. Based on observed wall clock time and script changes during those periods, a threshold of 60 minutes was identified for periods of inactivity, which were subsequently excluded from time on task (cf. Table 2 in Appendix B).

Executed Scripts. Executed scripts were saved as standalone versions together with their associated execution time and accessed netlist components. Each script execution represents an intermediate state of an iteratively developed solution script.

Manual Component Selections. Manual selections of netlist components together with the associated time were extracted.

Time per Phase. The phases of human sense-making as described in Section 2.2 were detected by checking the following conditions for the completion of each phase:
Phase 1 is completed when all 50 candidates are identified in the netlist. Thus, they have to be accessed by a script.
Phase 2 is finalized when all 30 targets which actually implement a watermark are identified and the watermarking signatures are read out. Therefore, data from those components has to be extracted and processed via script.
Phase 3 is finished after removing all watermark signatures from the circuit. This implies that all 30 targets have to be manipulated. Consequently, the script solving Phase 3 has to conduct a manipulation effectively removing the watermarking on those targets.

Fulfillment of those conditions was checked through a combination of automated script analysis, for example, when and

how relevant components are accessed, as well as further manual reviews of scripts. Manual reviews were conducted collaboratively by two researchers familiar with the task under study.

Progress Metric. As described in Section 2.1, reverse engineers use the two principal actions of semi-automated scripts and manual analysis. We assessed progress during the Study Task with respect to both actions.

Script Progress Score. Regarding semi-automated reverse engineering, we employed automated script analysis, e.g, the observation of persistent lines of code with respect to the final solution script, and further manual reviews of single script iterations. For each executed script, a progress score between 0 and 3 was assigned according to the rules below:

0: no progress made: no or only obsolete code added,
1: few relevant line(s) of code implemented; small subproblem solved,
2: relevant block of code or single significant line of code implemented; important subproblem solved,
3: significant idea or significant block of code implemented.

Manual Progress Score. With respect to manual analysis, groups of manual interactions were assessed on the same scale by first pooling consecutive component selections and then allocating the following progress scores between 0 and 3:

0: selection of irrelevant component(s),
1: repeated selection of a single relevant component,
2: repeated selection of multiple relevant components,
3: first selection of relevant component(s).

Progress Visualization. To enable visual representations of progress during HRE processes, we allocated weights to the progress scores. Progress Scores 0 and 1 were weighted as is, while Scores 2 and 3, which represent the main progress and occur only sparsely, were allocated a weight of 3 respectively 7. Those weighted scores served as foundation for the graphical progress visualization as follows: For each participant and phase, all assigned weighted scores were summed up. Progress made is represented as the fraction of the weighted progress score divided by the aforementioned sum.

4 Results

In the following section, we briefly illustrate the results of the expert and classify them in relation to the results of our eight participants before we examine the behavioral data and cognitive abilities of the participants with respect to the research questions formulated in Section 2.4. Section 4.2 provides insights for RQ1 by analyzing occurrence and relevance of phases from the three-phase model (cf. Section 2.2). With regard to RQ2, the respective strategies of the overall most and least efficient participants are evaluated in Section 4.3. Section 4.4 presents first indications about cognitive factors and HRE regarding RQ3.

4.1 Sanity Check with HRE Expert

The total solution time of the recruited expert for correctly solving the Study Task in HAL was 162 minutes, which is comparable to the fastest participant of our study (169 minutes). The occurrence of all three phases could clearly be detected via analysis of the behavioral data as described in Section 3.3 (cf. Figure 1). This is notable because the expert has an entirely different training background than the study participants. In Phase 1, the expert identified the watermark candidates and saved them in a data structure, before writing a function to successfully extract and functionally verify the watermark signatures, therefore identifying the targets. In the last phase, the expert removed the watermark signatures and generated the watermark-free netlist.

With 164 executed scripts, the expert lies within the upper range of our participants, who executed between 62 and 175 scripts (cf. Appendix B, Table 3). Also, the number of 204 manual component selections is comparable to our participants' (cf. Appendix B, Table 4).

Furthermore, the in-depth analysis of HRE strategies revealed that the expert applied strategies similar to those of the fastest participant (e.g., divide-and-conquer), and only rarely ran into periods of stagnation (cf. Figure 3 in Appendix B).

In summary, the expert's results and behavior are in line with the study participants, and therefore allow the assumption of classifying them as experienced reverse engineers. At the same time, the HRE expert solved the Study Task in a time comparable to the participants' time on task, which indicates the adequate difficulty of the task.

4.2 Empirical Observation of Phases (RQ1)

All participants were able to solve the Study Task correctly as indicated by solution probabilities between 97% and 100%. Based on the behavioral analysis described in Section 3.3, the occurrence of the three phases proposed in Section 2.2 could be observed for every participant. First, participants saved the watermark candidates in (different) data structures in Phase 1, before iterating over those candidates to apply their specific methods of functional verification to identify the targets in Phase 2. Only after successful verification, the participants began to remove the watermarking and to subsequently generate the watermark-free netlist in Phase 3. In the following passages, we report our results regarding spent time, executed scripts, and manually inspected netlist components per phase.

4.2.1 Solution Time per Phase

Figure 1 shows the time each participant required for solving the Study Task, together with a break down of the times they spent on each of the three phases. On average, participants spent 254 minutes ($SD = 74$ minutes) on task. We also calculated the group average of relative time spent on each of the three phases, by averaging the relative times of all eight participants: On average, they spent 12% ($SD = 4\%$) of their

time in Phase 1, Phase 2 consumes 58% ($SD = 11\%$), and Phase 3 took 30% ($SD = 10\%$) of participants' time solving the task.

4.2.2 Executed Scripts per Phase

On average, participants executed 117 scripts ($SD = 32$) while solving the Study Task. We calculated the relative number of executed scripts per phase by averaging relative number of scripts per participant: 15% ($SD = 9\%$) of scripts were executed in Phase 1, 55% ($SD = 13\%$) in Phase 2, and 30% ($SD = 12\%$) in Phase 3. The number of executed scripts per phase is represented in Appendix B, Table 3.

4.2.3 Manual Component Inspections per Phase

On average, participants manually inspected 123 ($SD = 82$) netlist components while working on the Study Task. We calculated relative numbers of inspected components per phase by averaging relative numbers per participant. Participants conducted 48% ($SD = 35\%$) of their manual inspections in Phase 1, 42% ($SD = 38\%$) in Phase 2, and 10% ($SD = 8\%$) in Phase 3. Table 4 in Appendix B shows detailed statistics on manual component inspections for all participants.

4.3 HRE Strategies (RQ2)

In this section, we perform an in-depth exploration of HRE strategies per phase for Participants 1 (P1) and 8 (P8), the least and most efficient overall participants with respect to solution time. While P1 required a total time of 398 minutes, P8 solved the task in 169 minutes. Both participants spent a similar amount of relative time in Phases 1 to 3 (cf. Figure 1), and had comparable total amounts of manual netlist interactions (P1: 127, P8: 90). P8 executed the least (62) and P1 the third-most (132) scripts. We want to illustrate that this was not a problem of poor programming skills: Whereas only 47% of the scripts written by P8 were syntactically correct, P1 executed 73% syntactically correct scripts. A detailed overview of the assigned progress scores per phase as introduced in Section 3.3 is shown in Appendix B, Table 5. In the case of executed scripts, a progress score of 0 — which indicates stagnation — dominates all phases but Phase 1 of P8. Manual analyses have crucial impact (score of 3) on progress in Phase 1 for both participants, and on Phase 2 for P8. Overall, script-based progress dominates progress made through manual analyses. A visualization of participants' P1 and P8 progress over time is shown in Figure 3 in Appendix B. The figure visually complements the exploration of strategies applied by P1 and P8 in Phases 1, 2 and 3 as presented below.

4.3.1 Phase 1 (Candidate Identification)

P1 required 54 minutes to solve Phase 1, while P8 identified the correct candidates in 20 minutes. Both participants started with manual analysis and were able to achieve initial progress by selecting relevant components: P1 detected the first relevant components after 7 minutes, and P8 found the first relevant components after 4 minutes. After initial progress had been made, P8 implemented his concept of structural candidate identification in the very first script (minute 18), which then only needed minor adjustments in order to finish Phase 1. P1 similarly implemented his crucial idea for Phase 1 in the first script (minute 26) but needed several script iterations in order to successfully identify candidates. Despite their similar solution strategies with respect to interactions of manual and script-based analyses, our manual script review revealed that P1 chose a less efficient and more complex implementation to detect the candidates as indicated by a larger search space and a deeper nesting of the algorithm.

4.3.2 Phase 2 (Candidate Verification)

Participants 1 and 8 spent 111 respectively 228 minutes in order to functionally verify the candidates. P8 made immediate progress through manual inspection of relevant components. Afterwards, P8 progressed towards the verification of candidates via the development of scripts and single manual inspections without significant periods of stagnation. Only between minutes 89 and 99, the participant could not advance the problem of data extraction from netlist components via script. In our manual reviews of scripts, we observed that P8 applied a divide-and-conquer strategy, and considered the overall objective — removing the watermark signatures — already in his approach of extracting them in Phase 2.

On the contrary, P1 started with an initial stagnation period (minute 54 to 78), where three scripts were executed without noticeable direction. His initial progress in Phase 2 was sparked through the manual inspection of relevant components, although there were still irrelevant components among the inspected. The next significant stagnation period (minute 131 to 203) concerned the same problem causing the 10-minute stagnation of P8. Another period of stagnation lasted from minute 210 to 242, where P1 inspected irrelevant components. Manual review of scripts revealed that P1 developed an efficient but complex algorithm for Phase 2 including bitwise binary processing depending on several conditions.

4.3.3 Phase 3 (Realization)

The realization of the underlying objective took 116 minutes for P1 and 37 minutes in the case of P8. Consequently, P8 moved quickly towards the solution without manual netlist inspections. In the stagnation period from minute 153 to 168, he merely tried to fix printing methods. P1 had a significant stagnation period from minute 305 to 352, where the manipulation of binary strings represented a significant hurdle. After this stagnation period, manual component inspections served as impulse for further progress. In the last stagnation period, P1 went back to the algorithm from Phase 1 to replace hard-coded data with variables.

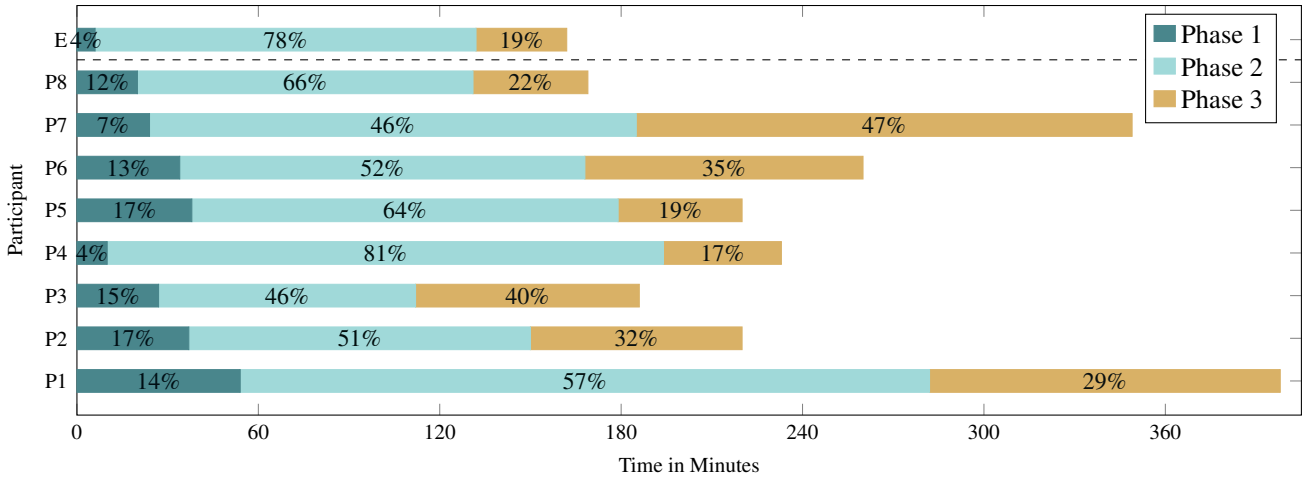

Figure 1: Absolute and relative solution times per participant and phase. For comparison, solution times of the expert are represented above the dotted line.

4.4 Cognitive Factors and HRE (RQ3)

The small sample size rendered statistical analyses of the influences of cognitive factors on HRE performance impossible. We do, however, want to briefly describe one interesting finding which can be discussed as an incentive for future research on cognitive factors in HRE. Our data suggest a potential negative correlation between Working Memory (WM) scores and the overall solution time of the Study Task. The descriptive data shows that participants with higher scores in WM tend to solve the task quicker than participants with lower WM scores. P8 had the highest WM score (126) and achieved the shortest time on task with 169 minutes. Meanwhile, the least efficient participant (P1) with a time on task of 398 minutes had a lower WM score of 108. There appears to be one outlier in the data (P6) which will be discussed in Section 5.3. The descriptive and visual analyses by scatter plots for both cognitive factors Processing Speed (PS) and Perceptual Reasoning (PR) did not show comparable results. Appendix B, Table 6 summarizes the descriptive data of cognitive factors and solution time.

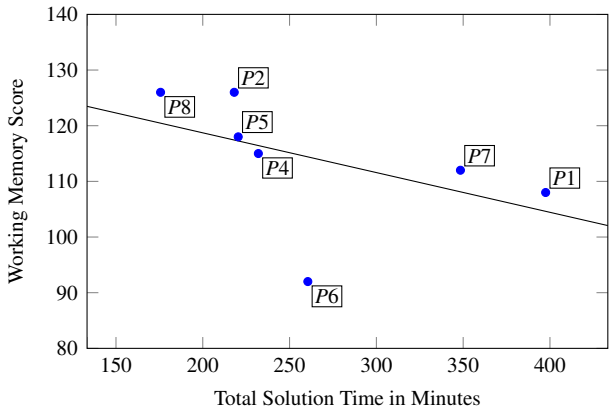

Figure 2: Scatter plot of solution time (x-axis) and Working Memory scores (y-axis) with trend line. P3 did not participate in the cognitive tests.

5 Discussion

In this section, we discuss implications of the results presented in Section 4 with respect to our four research questions.

5.1 HRE Phase Model (RQ1)

We were able to detect the proposed three-phase model for netlist reverse engineering for all participants of our study (cf. Figure 1). Thus, we suggest that passing through all phases in their respective order is essential to solve the underlying task. Our data revealed differences in absolute and relative times spent per phase, implicating their respective levels of difficulty. Moreover, we observed HRE as an interwoven process of manual and script-based netlist interactions. The observed proportions of manual component inspections per phase imply a high relevance of visual inspections for Phases 1 and 2. Even though amounts of manual interactions varied between participants as indicated by high standard deviations, we could observe purposeful usage for every participant. In the following, we briefly discuss our results with respect to each phase.

5.1.1 Phase 1 (Candidate Identification)

The relatively short solution times observed in this phase indicate that the identification of candidates, that is, relevant components, could be solved efficiently via structural analyses as described in Section 2.2. Since most manual netlist interactions were conducted in Phase 1, we deduce that they play an important role in order to detect and inspect starting

points (e.g., global in- and outputs, or nets with a great number of sinks) for reverse engineering. The identification of candidates in Phase 1 is a necessary precondition to conduct targeted candidate verification in Phase 2. It is therefore — despite its relatively short duration as observed in this exploratory study — crucial for the overall HRE process.

5.1.2 Phase 2 (Candidate Verification)

The significant portion of total solution time spent on Phase 2 implies that candidate verification was the most challenging subtask for our participants. In order to verify candidates, participants developed algorithms which functionally analyzed single candidates. We assume that the development of such an algorithm is a challenging problem. The results suggest that manual analyses in Phase 2 supported most participants in verifying candidates and developing their algorithms. A sound solution of Phase 2 is very important as preparation for the specific HRE objectives realized in Phase 3.

5.1.3 Phase 3 (Realization)

In Phase 3, reverse engineers conduct goal-oriented actions on previously identified targets by means such as interpretation, manipulation, simulation, or a combination thereof. In our case, watermarks had to be removed from the netlist via manipulation. Although netlist manipulation was an essential part of the training tasks, it was difficult to transfer those skills to the application of removing watermarks as indicated by a portion of 30% of total solution time. Participants applied manual interactions only sparsely in Phase 3, suggesting that they have less impact here compared to Phases 1 and 2.

5.2 HRE Strategies (RQ2)

In this section we discuss our results from the in-depth exploration of HRE strategies of the most and least efficient participants (cf. Section 4.3).

The comparable amounts of manual netlist interactions for P1 and P8 indicate their importance for HRE regardless of the reverse engineers' efficiency. Different amounts of executed scripts between both participants are an evidence that the faster participant made more progress per script. However, our data suggests that P8 was not the better programmer than P1, implying that other factors contribute to a higher level of progress per script.

With respect to our progress score assignments (cf. Table 5), we note that scores of 0 indicating stagnation dominate the overall HRE progress. This supports the assumed high complexity of netlist reverse engineering even for medium-sized netlists as in the underlying Study Task. The observation that script-based progress dominates progress made by manual analyses implies the nature of manual analysis methods as supporting factor for HRE.

In the following passages, we discuss the progress over time of P1 and P8 with special emphasis on periods of stagnation (cf. Figure 3 in Appendix B).

5.2.1 Phase 1 (Candidate Identification)

In Phase 1, both participants followed similar strategies based on structural netlist exploration via semi-automated scripts. The fact that initial progress was made through manual component inspections by both participants further supports the indication from Section 5.1 that manual inspections play an important role at early stages of netlist reverse engineering. Longer duration of Phase 1 for P1 was caused by a more complex algorithmic approach for candidate identification which is harder to implement.

5.2.2 Phase 2 (Candidate Verification)

Both participants followed different strategies in Phase 2 in order to verify candidates identified in Phase 1. P1 started with manual inspection of several relevant components, which then were manually divided into subgroups of candidates. Subsequently, P8 started — supported by further manual analysis — the development of an algorithm applying a divide-and-conquer strategy. Thus, we assume that manual analysis is a crucial factor for choosing this strategy and therefore to solve Phase 2 in a shorter amount of time. Towards the end of Phase 2, P8 applied his algorithm in a slightly adapted version to the other subproblems. Consequently, the divide-and-conquer strategy seems to be an efficient approach for solving HRE problems in this phase.

On the contrary, no clear strategy is identifiable for P1 initially. We observed that P1 tried to continue developing the candidate extraction algorithm from Phase 1 for this phase. We assume that P1 did not develop such a concrete plan as P8 based on the long-lasting stagnation periods observed throughout Phase 2.

Additionally, we monitored a significant stagnation period of 70 minutes, where P1 tried to read-out data from candidates; P8 solved the same problem within 10 minutes. We assume that P8 was able to apply prior knowledge from training tasks faster, since the they taught all necessary methods to solve this problem. A further stagnation period during a manual component selection could be observed for P1. We hypothesize that P1 tried to explore new points of interest in order to continue the verification of his candidates. This is a further hint that P1 applied a sub-optimum strategy.

5.2.3 Phase 3 (Realization)

Forward-thinking of P8 enabled the fast solution time in Phase 3 because the participant could adapt the existing algorithms from Phase 2 in order to remove the watermark. In contrast, P1 struggled as indicated by a long stagnation period in which the participant tried to remove the watermarking on a binary level. Additionally, P1 still needed visual input by manual netlist interactions in order to execute the planned strategy, which is a time-consuming process. In the last stagnation period, P1 tried to improve the algorithm for candidate identification originally implemented in Phase 1.

5.3 Cognitive Factors and HRE (RQ3)

In the light of RQ3, the conduction of statistical analyses was infeasible based on the small sample size. Nevertheless, we reported a promising result concerning a possible negative correlation between scores in Working Memory (WM) and solution time. The term WM is defined as a brain system which enables the storage of sensory input (e.g., visual input) in immediate awareness and the manipulation of that input in order to solve complex cognitive tasks like problem solving [2]. It consists of several parts, for example, the central executive (attentional-controlling system) [3]. Based on a descriptive analysis the data suggested that participants with higher WM scores tend to solve the task faster than participants with lower WM scores. Due to missing statistical analyses, we can only hypothesize that the WM might play a role in solving HRE problem tasks and future studies should statistically investigate the interplay between WM and the efficiency of solving HRE tasks.

Nevertheless, we try to explain differences in problem solving strategies of the participants by referring to the functionalities of the WM. Baddeley and Hitch postulated that the WM is central for solving cognitively difficult tasks like problem solving by performing mental operations on temporally stored information [4]. Against this background, it might be possible that P8 with the highest WM score achieved a more efficient solution than P1 who had a lower WM score. P8's strong WM might have supported the participant in mentally dividing the main problem into several sub problems, and helped in selecting further actions in order to achieve sub goals. Moreover, we assume that P8's stronger WM might have helped to conclude a stagnation period in Phase 2 more efficiently (10 minutes) by quicker activating relevant prior knowledge compared to P1 who needed over 70 minutes. Both participants struggled with the same problem in reading out relevant information from candidates. Although both 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, P8 was quicker in solving that problem. We hypothesize, that the stronger WM enabled P8 to activate stored information from the long-term memory in order to analyze and work with inputs the participant stored in immediate awareness in the WM. This ability of activating prior knowledge is described by the term chunks [7], which postulates that the working load of the WM can be reduced by activation of knowledge structures. Those chunks can boost the immediate memory of the WM and are connected to the level of expertise. A stronger WM could have been advantageous in producing a faster and more efficient solution.

Moreover, the scatter plot also revealed an outlier (P6) with the lowest WM score. A possible explanation for this outlier is that the measurements of WM abilities could have been influenced by uncontrolled variables. Prior work showed that the performances in WM tasks can be impaired by several variables, such as chronic psychological stress [22], acute psychological stress [26], negative emotions like anxiety [24], or neural disorders [44].

5.4 Hypotheses for Cognitive Obfuscation (RQ4)

Hardware obfuscation is understood as design method that impedes reverse engineering. Even though it should not be considered a silver bullet for hardware security, it can be a useful tool, for example, for improving IP protection or increasing the cost factor of an attack. Two important observations of our exploratory study are that (i) all participants progressed through a unique phase model and that (ii) the principal methods with respect to manual and script-based analyses employed by the participants were similar. This allows us to derive hypotheses for hardware obfuscation that take this process steps into account.

5.4.1 Obfuscation delaying HRE Phases

Our data showed that Phase 1, the necessary first step of netlist reverse engineering, can be solved efficiently. In order to increase time on Phase 1, we suggest the following approach for cognitive obfuscation: The netlist should be composed of many equally-looking regular structures, for example, by employing dummy wires or camouflaged gates [36]. Only few of those structures should contain the actual HRE targets. Thus, Phase 1 search techniques will return a considerable number of candidates. As a result Phase 1 will be slowed down through the amount of candidates found, as well as Phase 2 since all candidates have to be verified functionally.

5.4.2 Obfuscation impeding HRE Strategies

Hardware reverse engineers have to develop strategies that are tailored to their specific objective. From a problem solving perspective, Dörner and Funke concluded that a universal problem solving strategy which can be applied to solve different problems does not exist [10]. Against this background, it seems promising to apply different cognitive obfuscation methods for a given netlist which are similar in appearance but require individual solution strategies. This will force the reverse engineer to develop and apply various strategies rather than using a universal one.

5.4.3 Obfuscation against Cognitive Abilities

Based on our initial findings on the role of Working Memory (WM), it is possible that the performance in HRE might be correlated to levels of WM. If future studies can support this hypothesis with statistical data, the development of countermeasures could aim to overload cognitive capabilities of a reverse engineer. One important aspect in this context is the capacity of the WM. Even though the commonly assumed limit of seven items [21] can be extended by activating knowledge structures (chunks) stored in the long-term memory [7,11,16],

the capacity of items that can be stored will always be a rather moderate number. We hypothesize that cognitive obfuscation could overload the capacity of the WM. A possible approach might be the development of structures which force the analyst to apply a combination of many different strategies simultaneously (e.g., simulation in Phase 3 in combination with structural and functional analyses in Phases 1 and 2) which might overload the capacity of the WM.

5.5 Limitations and Future Work

While our exploratory study provides important initial insights into the technical and cognitive processes underlying HRE, there are certainly limitations. First, this research is limited by the small sample size which did not allow us to conduct any statistical analyses. With respect to cognitive factors, future research with larger sample sizes could quantify the influences of WM on HRE processes more accurately. In this context, it would also be interesting to explore to what extent prior knowledge and chunks are activated during HRE.

Second, our results are based on the behavioral analyses of cyber security students who acquired relevant HRE skills and knowledge during an extensive training phase. Even though we worked with students, we took precautions to compare our sample and the Study Task by recruiting an HRE expert. The descriptive analyses of this expert served as a sanity check for our findings, the approximation of experienced revers engineers with our student sample, and the appropriate difficulty of the Study Task. Our results, discussion and takeaways regarding strategies were obtained analyzing two participants. Further detailed analyses of additional participants' and of the expert's HRE problem solving strategies would have been worthwhile. However, given that the primary goal was to provide first insights into the technical and cognitive processes of HRE, we feel that our analysis is warranted and led to interesting results worth reporting. We hope that our exploratory study will trigger follow-up work which focuses on the in-depth analysis of problem solving strategies in HRE. In the future, qualitative data such as interview data may complement the comprehension of technical and cognitive processes in HRE and could potentially lead to a more nuanced view of the strategies the participants followed.

Third, we developed our HRE model based on the behavioral analyses of reverse engineers for a single, medium-complex HRE task and it remains unclear whether the phases transfer to different HRE settings. Nevertheless, it seems plausible that the three phases can be found in other HRE tasks too, but more complex real-world tasks could potentially lead to an extension of the phase model, for example, by discovering further sub phases that are universally observed.

Finally, we proposed first hypotheses for cognitive obfuscation techniques. Future work should evaluate which cognitively difficult tasks are suitable to raise time on HRE tasks and how they can be implemented in netlists.

6 Conclusion

The motivation behind this work is to take a first step towards a better understanding of technical and cognitive processes in HRE, an area with little prior published work despite its importance in commercial and national security contexts. We conducted an exploratory behavioral study with eight participants who solved a realistic HRE task. We examined the approximation of the participants as experienced reverse engineers and the appropriate difficulty of the Study Task through behavioral analyses of an HRE expert. We found that the three-phase model for HRE, which we had postulated, was in fact executed by all participants (as well as by the expert). The analysis of behavioral data showed that the two principal types of actions, manual and automatic analyses, are closely interwoven during a given HRE task. The analysis of cognitive prerequisites of the participants indicates that the Working Memory (WM) might play an important role in solving HRE problems.

Our study suggests several new research directions that can be worthwhile to explore. It seems promising to extend the study to a larger population of participants and, in particular, to observe true HRE experts using a similar methodology as presented here. Another especially interesting research direction lies in the design of a novel class of countermeasures against HRE, which will take cognitive limitations of reverse engineers into account.

Acknowledgements

We would like to thank Yasemin Acar, Maximillian Golla, as well as our anonymous shepherd and all reviewers for their constructive feedback on this work. We are also grateful to Oliver Gebhard and René Walendy for supporting us with data analysis. Finally, we thank Carl for contributing the "Carlgorithm" at an early stage of this research.

This work was supported in part by DFG Excellence Strategy grant 39078197 (EXC 2092, CASA), through ERC grant 695022 and NSF grant CNS-1563829.

References

[1] Leonid Azriel, Ran Ginosar, and Avi Mendelson. SoK: An Overview of Algorithmic Methods in IC Reverse Engineering. In *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, pages 65–74, 2019.

[2] Alan Baddeley. Working Memory. *Science*, 255(5044):556–559, 1992.

[3] Alan Baddeley. Working Memory: Theories, Models, and Controversies. *Annu. Review of Psychology.*, 63:1–29, 2012.

[4] Alan Baddeley and Graham Hitch. Working Memory. In *Psychology of Learning and Motivation*, volume 8, pages 47–89. Elsevier, 1974.

[5] Jens F Beckmann and Natassia Goode. The Benefit of Being Naïve and Knowing It: The Unfavourable Impact of Perceived Context Familiarity on Learning in Complex Problem Solving Tasks. *Instructional Science*, 42(2):271–290, 2014.

[6] R. S. Chakraborty et al. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Trans. CAD of Integrated Circuits and Systems*, 28(10):1493–1502, 2009.

[7] William G Chase and Herbert A Simon. Perception in Chess. *Cognitive Psychology*, 4(1):55–81, 1973.

[8] Gregory H Chisholm, Steven T Eckmann, Christopher M Lain, and Robert L Veroff. Understanding Integrated Circuits. *IEEE Design & Test of Computers*, 16(2):26–37, 1999.

[9] Z. Ding et al. Deriving an NCD file from an FPGA bitstream: Methodology, architecture and evaluation. *Microprocessors and Microsystems*, 37(3):299–312, 2013.

[10] Dietrich Dörner and Joachim Funke. Complex Problem Solving: What It Is and What It Is Not. *Frontiers in Psychology*, 8, 2017.

[11] K Anders Ericsson and Walter Kintsch. Long-Term Working Memory. *Psychological Review*, 102(2):211–245, 1995.

[12] Andreas Fischer, Samuel Greiff, and Joachim Funke. The Process of Solving Complex Problems. *Journal of Problem Solving*, 4(1):19–42, 2011.

[13] Marc Fyrbiak, Sebastian Strauß, Christian Kison, Sebastian Wallat, Malte Elson, Nikol Rummel, and Christof Paar. Hardware Reverse Engineering: Overview and Open Challenges. In *2017 IEEE 2nd International Verification and Security Workshop*, pages 88–94. IEEE, 2017.

[14] Marc Fyrbiak, Sebastian Wallat, Jonathan Déchelotte, Nils Albartus, Sinan Böcker, Russell Tessier, and Christof Paar. On the Difficulty of FSM-based Hardware Obfuscation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 293–330, 2018.

[15] Marc Fyrbiak, Sebastian Wallat, Pawel Swierczynski, Max Hoffmann, Sebastian Hoppach, Matthias Wilhelm, Tobias Weidlich, Russell Tessier, and Christof Paar. 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, 2018.

[16] Fernand Gobet and Herbert A Simon. Templates in Chess Memory: A Mechanism for Recalling Several Boards. *Cognitive Psychology*, 31(1):1–40, 1996.

[17] David Z Hambrick and Randall W Engle. The Role of Working Memory in Problem Solving. *The Psychology of Problem Solving*, pages 176–206, 2003.

[18] Jill Larkin, John McDermott, Dorothea P Simon, and Herbert A Simon. Expert and Novice Performance in Solving Physics Problems. *Science*, 208(4450):1335–1342, 1980.

[19] NY Louis Lee and PN Johnson-Laird. A theory of reverse engineering and its application to Boolean systems. *Journal of Cognitive Psychology*, 25(4):365–389, 2013.

[20] Ewen MacAskill and Gabriel Dance. NSA Files: Decoded. https://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded, November 2013. [Online; accessed 2020-February-26].

[21] George A Miller. The Magical Number Seven, Plus or Minus Two Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63(2):81, 1956.

[22] Kazushige Mizoguchi, Mitsutoshi Yuzurihara, Atsushi Ishige, Hiroshi Sasaki, De-Hua Chui, and Takeshi Tabira. Chronic Stress Induces Impairment of Spatial Working Memory Because of Prefrontal Dopaminergic Dysfunction. *Journal of Neuroscience*, 20(4):1568–1574, 2000.

[23] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks: Extracting Keys from Xilinx Virtex-II FPGAs. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 111–124, 2011.

[24] Tim P Moran. Anxiety and Working Memory Capacity: A Meta-Analysis and Narrative Review. *Psychological Bulletin*, 142(8):831, 2016.

[25] Allen Newell and Herbert Alexander Simon. *Human Problem Solving*, volume 104. Prentice Hall Englewood Cliffs, NJ, 1972.

[26] Shaozheng Qin, Erno J Hermans, Hein JF van Marle, Jing Luo, and Guillén Fernández. Acute Psychological Stress Reduces Working Memory-Related Activity in the Dorsolateral Prefrontal Cortex. *Biological Psychiatry*, 66(1):25–32, 2009.

[27] Shahed E Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, and Mark Tehranipoor. A Survey on Chip to System Reverse Engineering. *ACM Journal on Emerging Technologies in Computing Systems*, 13(1):6, 2016.

[28] Michael G Rekoff. On Reverse Engineering. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 244–252, 1985.

[29] Jordan Robertson and Michael Riley. The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies. https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies, October 2018. [Online; accessed 2020-February-26].

[30] Adam Satariano. Huawei Tells Parliament It's No Security Threat, Aiming to Avoid a Ban. https://www.nytimes.com/2019/06/10/technology/huawei-britain-parliament-ban.html, Jun 2019. [Online; accessed 2020-February-26].

[31] Moritz Schmid, Daniel Ziener, and Jürgen Teich. Netlist-Level IP Protection by Watermarking for LUT-Based FPGAs. In *2008 International Conference on Field-Programmable Technology*, pages 209–216. IEEE, 2008.

[32] Pramod Subramanyan, Nestan Tsiskaridze, Wenchao Li, Adria Gascon, Wei Yang Tan, Ashish Tiwari, Natarajan Shankar, Sanjit A Seshia, and Sharad Malik. Reverse Engineering Digital Circuits Using Structural and Functional Analyses. *IEEE Transactions on Emerging Topics in Computing*, 2(1):63–80, 2014.

[33] M Tehranipoor and F Koushanfar. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design & Test of Computers*, 27(1):10–25, 2010.

[34] Texplained. Chipjuice IC Reverse Engineering Software. https://www.texplained.com/about-us/chipjuice-software/. [Online; accessed 2020-February-26].

[35] R. Torrance and D. James. The State-of-the-Art in IC Reverse Engineering. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 363–381. Springer, 2009.

[36] Arunkumar Vijayakumar, Vinay C Patil, Daniel E Holcomb, Christof Paar, and Sandip Kundu. Physical Design Obfuscation of Hardware: A Comprehensive Investigation of Device-and Logic-Level Techniques. *IEEE Transactions on Information Forensics and Security*, 12(1):64–77, 2016.

[37] Daniel Votipka, Seth M Rabin, Kristopher Micinski, Jeffrey S Foster, and Michelle L Mazurek. An Observational Investigation of Reverse Engineers' Processes. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, August 2020. USENIX Association.

[38] Sebastian Wallat, Nils Albartus, Steffen Becker, Max Hoffmann, Maik Ender, Marc Fyrbiak, Adrian Drees, Sebastian Maaßen, and Christof Paar. Highway to HAL: open-sourcing the first extendable gate-level netlist reverse engineering framework. In *Proceedings of the 16th ACM International Conference on Computing Frontiers - CF '19*, pages 392–397. ACM Press, 2019.

[39] Sebastian Wallat, Marc Fyrbiak, Moritz Schlögel, and Christof Paar. A Look at the Dark Side of Hardware Reverse Engineering – A Case Study. In *2017 IEEE 2nd International Verification and Security Workshop*, pages 95–100. IEEE, 2017.

[40] David Wechsler. Wechsler Adult Intelligence Scale Fourth Edition (WAIS–IV). *San Antonio, TX: NCS Pearson*, 22:498, 2008.

[41] Carina Wiesen, Nils Albartus, Max Hoffmann, Steffen Becker, Sebastian Wallat, Marc Fyrbiak, Nikol Rummel, and Christof Paar. Towards Cognitive Obfuscation: Impeding Hardware Reverse Engineering Based on Psychological Insights. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 104–111. ACM, 2019.

[42] Carina Wiesen, Steffen Becker, Nils Albartus, Christof Paar, and Nikol Rummel. Promoting the Acquisition of Hardware Reverse Engineering Skills. *IEEE Frontiers in Education*, 2019.

[43] Carina Wiesen, Steffen Becker, Marc Fyrbiak, Nils Albartus, Malte Elson, Nikol Rummel, and Christof Paar. Teaching Hardware Reverse Engineering: Educational Guidelines and Practical Insights. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, pages 438–445. IEEE, 2018.

[44] Erik G Willcutt, Alysa E Doyle, Joel T Nigg, Stephen V Faraone, and Bruce F Pennington. Validity of the Executive Function Theory of Attention-Deficit/Hyperactivity Disorder: A Meta-Analytic Review. *Biological Psychiatry*, 57(11):1336–1346, 2005.

Appendix

A Survey Instruments

Participant Questionnaire

1. Please enter your pseudonym: _____

2. How old are you? Please indicate your age in years. _____
3. What is your target degree? Please select one answer.
   - ○ Bachelor of Science
   - ○ Master of Science
   - ○ Other: _____
   - ○ Prefer not to answer
4. What is your major? Please select one answer.
   - ○ Cyber Security
   - ○ Electrical Engineering
   - ○ Computer Science
   - ○ Other: _____
   - ○ Prefer not to answer
5. Please indicate the number of semesters you have studied so far. _____

Expert Questionnaire

1. How old are you? Please indicate your age in years: _____
2. What is your highest level of education? Please select one answer.
   - ○ Bachelor of Science
   - ○ Master of Science
   - ○ Bachelor of Arts
   - ○ Master of Arts
   - ○ Ph.D.
   - ○ Other: _____
3. What is your current job position? What are your responsibilities? _____
4. On a scale 1 to 5, how would you asses your hardware reverse engineering skill level? Please indicate your skill level on the 5-point scale from 1 (being a beginner) and 5 (being an expert) by choosing one answer.
   ○ 1 (Beginner) ○ 2 ○ 3 ○ 4 ○ 5 (Expert)
5. How many total years of experience do you have with hardware reverse engineering? Please indicate your answer in years. _____

B Full Results

Table 2: Amount of excluded inactivity periods over 60 minutes per participant and phase.

	P1	P2	P3	P4	P5	P6	P7	**P8**	**E**
Phase 1	0	1	0	0	0	0	0	0	0
Phase 2	2	0	0	2	1	0	1	0	1
Phase 3	3	0	3	0	0	1	1	1	0
Total	**5**	1	3	2	1	1	2	**1**	**1**

Table 3: Amount of executed scripts per participant and phase. For comparison, the experts' amount of executed scripts is represented in the rightmost column.

	P1	P2	P3	P4	P5	P6	P7	**P8**	**E**
Phase 1	28	21	33	3	42	20	4	3	5
Phase 2	60	51	72	111	103	35	49	39	122
Phase 3	44	27	21	23	30	40	57	16	37
Total	**132**	99	126	137	175	95	110	**62**	**164**

Table 4: Amount of manual component inspections per participant and phase. For comparison, the experts' amount of manual inspections is shown in the rightmost column.

	P1	P2	P3	P4	P5	P6	P7	**P8**	**E**
Phase 1	93	66	0	4	217	16	47	23	8
Phase 2	19	4	37	162	8	0	93	67	194
Phase 3	15	0	1	21	59	4	24	0	2
Total	**127**	70	38	187	284	20	164	**90**	**204**

Table 5: Overview of assigned progress scores for executed scripts and manual analysis assigned to P1, P8, and the expert as explained in Section 3.3. Dashes indicate zero occurrences.

		P1				P8				Expert			
	Score	0	1	2	3	0	1	2	3	0	1	2	3
Phase 1	Script	22	3	3	1	1	-	1	1	1	2	1	1
	Manual	1	-	-	1	1	-	-	1	8	-	-	-
Phase 2	Script	48	7	4	-	29	5	2	2	92	26	5	1
	Manual	7	2	-	-	1	3	-	1	5	6	2	1
Phase 3	Script	36	5	2	-	12	3	1	-	25	7	4	1
	Manual	2	3	-	-	-	-	-	-	-	1	-	-

Table 6: Scores of the cognitive factors Working Memory (WM), Processing Speed (PS) and Perceptual Reasoning (PR) per participant and time spent on the Study Task in minutes (P3 did not participate in the cognitive tests).

	P1	P2	P3	P4	P5	P6	P7	P8
WM	**108**	126		115	118	92	112	**126**
PS	106	146		146	119	109	119	117
PR	104	129		100	115	100	104	106
Time	**398**	218	186	232	220	261	348	**176**

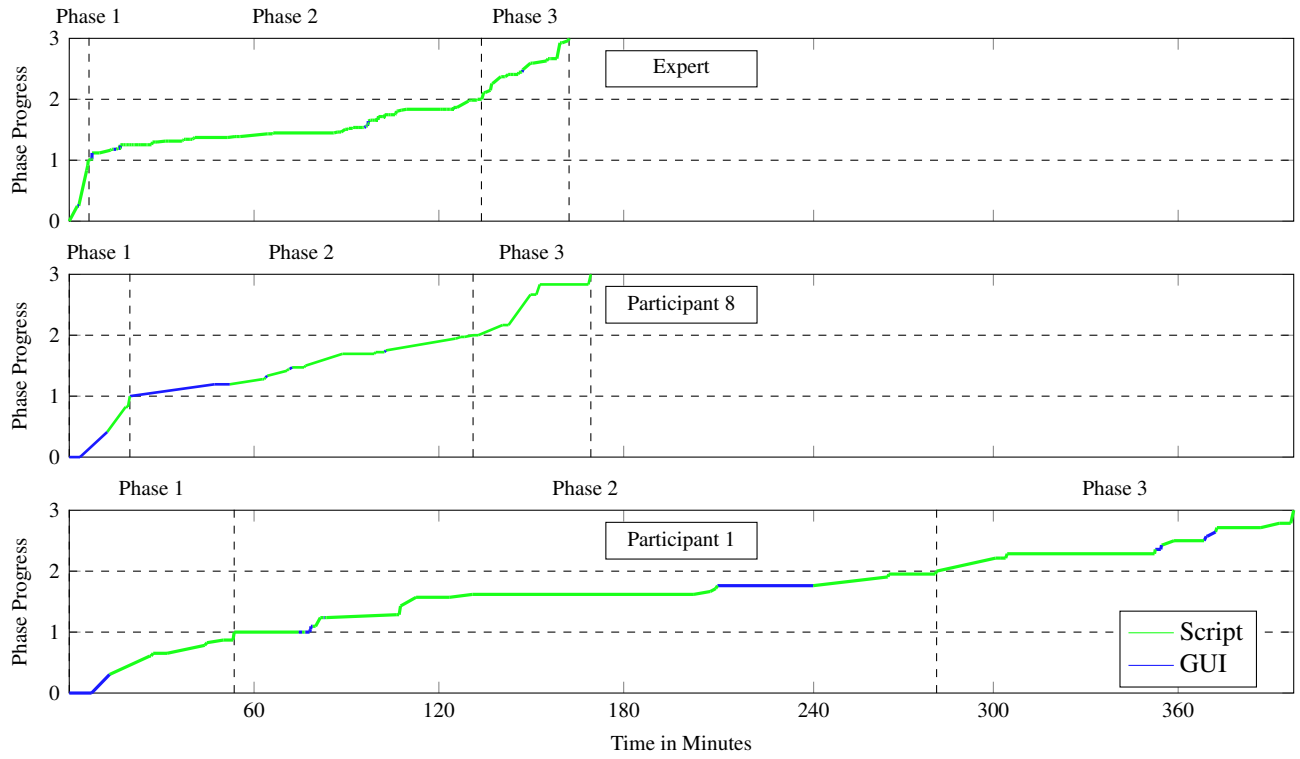

Figure 3: Progress visualization of the expert (top), P8 (middle), and P1 (bottom). The *x*-axis shows time in minutes, and the *y*-axis measures progress based on the progress metric described in Section 3.3. Blue lines indicate manual interactions with the netlist, while green lines indicate interactions with the netlist via semi-automated scripts. Reaching the horizontal lines 1, 2, 3 marks completion of the respective phases, which are also separated by vertical dotted lines.