# Improving Student Learning in Hardware Security: Project Vision, Overview, and Experiences

Robert Karam, Srinivas Katkoori, Mehran Mozaffari-Kermani
University of South Florida, Tampa, FL, USA
{rkaram, katkoori, mehran2}@usf.edu

***Abstract*—Practical, hands-on experience is an essential component of computer science and engineering education, especially in the cybersecurity domain. In this project, we are investigating techniques for improving student learning in such courses, first by developing a new hands-on hardware security course, then by testing the impact of *gamification* on student learning. The experiments utilize only inexpensive, open-source or freely-available software and hardware, and upon project completion, the modules themselves will also be made freely available online. Improving student learning in this critical area can have a widespread positive societal impact as we encourage students to have a security-first, secure-by-design mindset.**

## I. Introduction

Computers have become ingrained in nearly every aspect of modern life. Security of these systems has understandably been prioritized, with numerous national programs spurring research and development activity over the past few decades in cybersecurity. In turn, the rise in cybercrime has led to an increased need for cybersecurity professionals; universities have developed programs to meet that need through both traditional and online degree programs focusing on information and network security. Meanwhile, hardware, which supports and enables the processing required for more abstract systems, has been traditionally viewed as inherently trustworthy. In recent years, this perspective has shifted to one that rightly views hardware as a potentially vulnerable computing foundation: *unless the security of the underlying hardware can be guaranteed, any software or networking applications built on that platform would be equally vulnerable*.

To this end, many academic programs have introduced some form of hardware-oriented security curriculum. These courses have much in common, and generally cover supply chain security, testing and verification, hardware security primitives such as physical one-way functions (e.g. PUFs) or true random number generators (TRNGs), and invasive/semi-invasive/side-channel attacks and countermeasures. At the University of South Florida, we have previously offered hardware security course and have received very positive feedback from students about course material. However, much of the material is theoretical, with little practical or hands-on work. Some assignments require the use of datasets containing premeasured values from real hardware, and students are asked to analyze the data, build ML models, or similar coding tasks.

This material is based upon work supported by the National Science Foundation under Grant No. DGE-1954259. Contact: rkaram@usf.edu

Although these assignments served to ground some of the more theoretical aspects of the course material, they did not require any interaction with real hardware. Certainly, some topics naturally lend themselves to more theoretical foundations, such as supply chain security issues, while other topics are not amenable for a laboratory course at most universities due to their reliance on equipment that is costly, inaccessible, or requires significant training to operate safely and properly. ***Ensuring accessibility to the tools and materials required for teaching these topics is critical; basing hardware security education on inaccessible equipment can dramatically limit the number of students receiving such training in universities around the world.*** To enable not only our students, but students at universities across the world to access hands-on hardware security education, we developed a new course covering topics that 1) require only "accessible" equipment, broadly defined as equipment that is relatively inexpensive, open-source, and easy to use and operate; and 2) provide an opportunity to learn transferable hands-on hardware skills.

Education research is not simply about developing new courses or integration of research into existing courses. Of equal importance is researching the impact of pedagogy on student learning outcomes (SLOs). Hence, we have not only developed the new courses, we have developed a plan to investigate the impact of *gamification* on a hardware laboratory course. ***We hypothesized that game-based learning in a laboratory setting can improve SLOs, including skills such as proficiency with lab and measurement equipment and software / computer aided design (CAD) toolflows, as well as critical/creative thinking and brainstorming.*** To test this hypothesis, we designed course modules that were also amenable to team-based and collaborative learning through **gamification**. While gamification has been investigated as a tool to improve SLOs, to our knowledge it has not been studied in the context of a **hardware laboratory course**.

Broadly, *gamification* refers to the introduction of game mechanics or game-like features into non-game contexts [1]. Mechanics such as "race against the clock", and motivators such as "leaderboards" with point/scoring systems or badges/trophies/awards for completing certain milestones/tasks may be applied to hardware laboratory course modules. We believe that if some labs include a form of competition, students will be more likely to engage in the course, which will in turn have a positive impact on student learning of this complex subject area, and increased proficiency and comfort in using the lab equipment. In this paper, we will

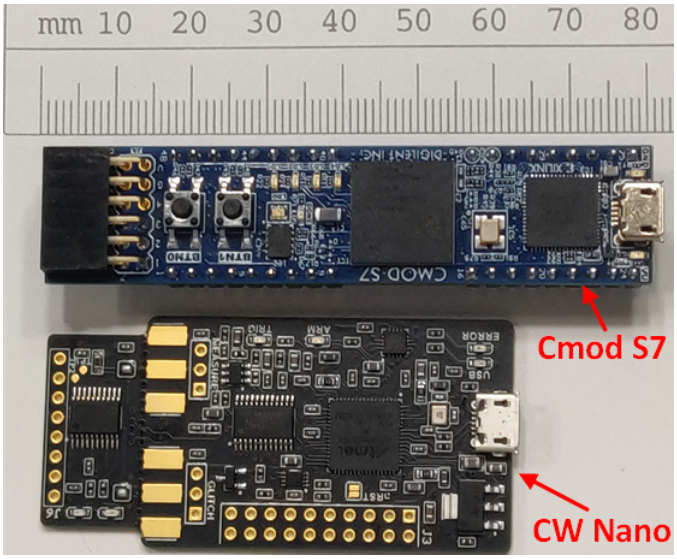


Fig. 1. The Digilent Cmod S7 and NewAE ChipWhisperer Nano boards selected for use in our hardware security laboratory course [2].

give a brief overview of the project, including a description of the course modules, hardware/software platform, evaluation plan, and preliminary results.

## II. Course Overview

### A. Description of Experiments

We designed nine hands-on experiments for this course. Each lab involves the use of various tools, either software only, hardware only, or a mix of both. The topics and related software/hardware tools are listed in Table I. This is suitable for computer engineering, electrical engineering, and computer science students as they gain exposure to many different facets of secure hardware design, including FPGA development, scripting languages like Python, and embedded software development in C/C++. We utilize two development boards, the Digilent Cmod S7 (FPGA) the ChipWhisperer (CW) Nano (microcontroller with support for side channel analysis experiments). Additional details on these boards are provided in Section II-B. Each experiment consists of a guided in-class portion designed to not only introduce students to the theory, but also serve as a tutorial or walk-through for the related experiment. Whenever possible, we aimed to have students work together, including requiring teammates to exchange data acquired from their own boards for analysis and evaluation.

### B. Hardware and Software Platform

In order to maximize the impact of this course development and education research, early in the project, we decided to use hardware platforms and software tools that are **open source and/or freely available**. We kept the following requirements in mind while choosing the lab hardware:

- *Low cost:* is important as it will help in assembling several lab kits at an affordable total cost so that each student has a kit of their own; it is also helpful in replacing broken hardware inexpensively
- *Portability:* Ideally, we would like the students to take the hardware home so that they have the freedom to work anywhere/anytime; further they can also bring to the classroom as needed
- *Open-source Software:* readily available for free and preferably maintained by the community will help in keeping the cost low and provide support through community forums

Two hardware boards satisfied the above requirements, namely, the Digilent Cmod S7 and the ChipWhisperer Nano (see Fig. 1). Digilent board dimensions are 79 mm x 18 mm x 18 mm. CW Nano's dimensions are 60 mm x 30 mm x 3 mm. Each can be powered by the same micro USB cable, is portable, and has very few external components. As these boards are not used at the same time, the same cable can be reused for both boards. CMod S7 can be programmed by student version (free) of Xilinx Vivado (Linux/Windows). A Linux based Virtual Machine (VM) can be used by Mac users to run the software. ChipWhisperer software is deployed as a VM image to simplify the software deployment for the class. We used VirtualBox (free for Windows, Mac, and Linux platforms) with no compatibility issues. VM deployment can cut down the support time (in-person or remote) which otherwise is significant if students were to install on their own. ChipWhisperer required installation of Python with support libraries, C compiler, bash shell, etc.

Digilent Cmod S7 has two (2) push buttons, four (4) LEDs, one (1) RGB LED, and a (small) expansion header. If additional user I/O is needed (say additional buttons/switches, displays, etc.), then S7 is breadboardable. S7 has Xilinx Spartan 7 FPGA with adequate logic block capacity needed for the labs we have developed. Through micro USB students can debug and observe the outputs from the mapped FPGA design through a Python script and the pySerial library. This is very convenient as it does not require memory initialization and sampling with a logic analyzer or an oscilloscope. S7 board supports labs 1, 3, and 7. Labs 1 and 3 cannot be simulated as they require real-world variations (process variations) in the hardware, while lab 7 can be simulated.

ChipWhisperer (CW) Nano houses two processors; one of them can be programmed with Python API and Jupyter notebooks and the other processor can be programmed with C to emulate a victim. Dozens of practical experiments on side-channel analysis can be developed on the CW board. The main Python-controlled acquision board can emulate a USB oscilloscope to record in real-time the victim's power consumption as the board executes various functions. The victim can be controlled with serial protocol so that the students can write, flash, and run various functions while recording, observing, analyzing, and plotting the results in real-time from Jupyter notebook environment. CW supports labs 4, 5, 6, and 8. For all labs, the students can analyze prior acquired data without access to the hardware. However, to learn how to use the hardware, troubleshoot and problem-solving on their own is a key outcome. Further, simply providing students with pre-recorded data will not allow students to make changes (e.g., implement a countermeasure against an attack) to the firmware and observe how the changes impact the power traces

TABLE I
OVERVIEW OF THE HARDWARE SECURITY LABORATORY EXPERIMENTS AND RELEVANT LANGUAGES / TOOLS.

| Lab | Name | Tools and Components |
|---|---|---|
| 1 | Physical Unclonable Functions (PUFs) Implementation and Evaluation | FPGA, Verilog, Python |
| 2 | Deep Learning and Modeling Attacks on Arbiter PUFs | Python |
| 3 | Implementation and Evaluation of Pseudo- and True Randomness in Hardware | FPGA, Verilog, Python |
| 4 | Introduction to Side Channel Analysis and Leakage Assessment | ChipWhisperer, Python, C |
| 5 | Correlation Power Analysis Attack and Countermeasure for AES | ChipWhisperer, Python, C |
| 6 | Timing Attacks and Countermeasures for Password Protected Embedded Systems | ChipWhisperer, Python, C |
| 7 | Hardware Trojan Design and Countermeasures in a Soft CPU | Verilog, RISC-V Assembly |
| 8 | Security for the Internet of Things: Attack and Countermeasure for RSA | ChipWhisperer, Python, C |
| 9 | Post-Quantum Cryptographic Hardware and Embedded Systems | Verilog |

(and success of the original attack). This necessitates physical access to the hardware by the students for data collection.

*By relying solely on COTS components and open-source hardware/software, we ensure that the results of this research may be replicated at other universities at relatively little cost*, especially compared with multi-million dollar equipment installations at other universities.

### C. Challenges in Course Preparation

Development of the course modules required careful consideration, not only to ensure full coverage of the topics for the learning objectives, but also to ensure all students would be able to run all of the required software. In particular, the FPGA board requires software for programming the bitstream. The complete Vivado package includes a device programmer, but the installation is large (70+ GB is typical) and the software may not run well on some student's laptops. An alternative standalone programmer such as Digilent Adept could be used, but this is only released for Windows and Linux. For the ChipWhisperer software, there is a complete software suite including a Python installation, Jupyter notebooks, various packages like numpy, matplotlib, other packages required for running the CW experiments, as well as gcc for compiling c programs, and drivers for communicating with the board. All of these separate components can be installed individually on a system, but to simplify the process for students, we assembled a Linux-based virtual machine (Ubuntu) pre-configured with all required software for programming the FPGA and configuring/running the CW experiments. Each experiment was set up as a Jupyter notebook. A "tutorial" portion of the experiment walks students through the basic techniques. Code examples are provided, and students can execute those samples, modify, and observe the effect, e.g. changes in data output from the FPGA or CW, or changes in power consumption from the CW. This approache ensures all students have a solid baseline knowledge of the techniques needed to successfully attack/defend, or implement and evaluate the effectiveness of their own attacks or countermeasures,

## III. METHODOLOGY

The project aims are twofold: 1) course development and 2) hardware security education research. In this section, we describe the education research component, which is presently on-going at the time of publication. This consists of teaching the class two ways - first, a more traditional lecture, with some in-class activities/demonstrations of hardware attacks, and second, a *gamified* version of the course. The experiment is constructed as an ABAB study, such that the gamification takes place in the "B" semesters. Two surveys, along with tracking metrics like grade averages and skill development, are used to assess the impact of gamification on student learning objectives.

### A. Hardware Skills Inventory

We begin by assessing the student's skills with a survey that collects demographic data, a self-assessment of computer engineering-related skills, and information on students' habits with respect to video games - how frequently they play games, and what kinds of games they play. The self-assessment asks students to list which topics they have a background in. This list includes the following topics: computer logic design, FPGA / reconfigurable computing, hardware security, software security / secure coding, network and information security, computer architecture, cryptography / finite fields, reverse engineering (hardware), reverse engineering (software), electronic circuit theory, transistors / nanoscale devices, and side channel analysis attacks. Following this, students are asked to rate their knowledge of, or comfort level using the following: mixed signal / digital oscilloscopes, logic analyzers, EM / current probes, multimeters, function generators, digital power supplies, matlab, C/C++, Java / C, Python, machine learning / AI, GPGPU, cryptography (theory), cryptography (implementation), Arduino/Raspberry PI or other single board computers, FPGAs, Verilog/VHDL, Synopsys or Cadence EDA tools, SPICE / circuit simulation, soldering / hardware prototyping, printed circuit board design, 3D modeling / 3D printing, and Git or other version control systems. Students are asked to rate their skills from 1 to 5, with 1 being no knowledge of or experience in the topic, and 5 being an expert on the topic. Finally, we collect data on student's extracurricular activities, whether or not they work on "side projects" outside of regular classwork, particularly ones related to computers/programming/hardware development. Examples given include, but are not limited to, developing a smartphone app or game, or building an Arduino-based device. We also ask students about their video game habits, whether or not they play games, how often, and what kinds.

In general, we expect that in the self assessment, students will rate themselves as being more familiar with topics and

tools covered in the course in the second survey compared to the first. We hypothesize that the amount this increases will be higher in "B" semesters compared to "A" semesters due to the differences in teaching methodology. Moreover, we are interested to determine if a correlation exists between the rate of improvement in the self-assessment in "B" semesters for students who routinely play video games compared to those that do not.

### B. Plans for Gamification

Research on gamification in higher education is relatively new, and coincides with the adoption and spread of video games. Indeed, numerous studies have noted a significant uptick in interest in gamification research in the past decade [3]–[5]. Preliminary results on the impact of gamification on learning have been generally mixed, with some authors finding positive trends, negative trends, or no apparent correlation. A recent meta-analysis by Hamari et al found a generally positive relationship between course module gamification and student engagement and learning, though they note the level of impact depends heavily on the context in which it is applied, as well as on the users themselves [3]. Borges et al further classified the extant literature on gamification and classified studies into those seeking to improve students' mastery of certain technical skills, challenge students with difficult problems, increase engagement, improve learning, cause behavioral change, or increase socialization [4], but no consensus was found on the result of these diverse studies.

Gamification has been investigated in a number of contexts related to computing, including introductory computer programming classes [6], IT compliance training [7], and cybersecurity competitions [8], among many others. One specific area that has received significant attention is in employing gamification to encourage students to engage in a "skills" type lab course, where students learn to use new lab equipment. The majority of literature on gamification in this context focuses on science lab courses, such as microbiology labs or similar [9]–[11]. Drace argues that one reason instructors of such courses have sought alternative pedagogical techniques is that, in a traditional classroom setting, a "skills" course may be perceived by students as a "task list" of unrelated assignments they must complete for a grade [10], and that alternative methods may provide a more coherent framework for increasing student engagement.

Our plan for gamification in this hardware security course generally involves adding some form of competition, either between student teams, or for individual students against a clock. When competing against other student teams, groups will practice an attack, implement a countermeasure, then attempt to mount an attack on another group's secured implementation. Beyond this, we plan to have a leaderboard in which individual teams will be ranked based on their ability to break another implementation. For example, if a lab involves a side channel power analysis attack, and Team A is able to extract an encryption key from the secured implementation using 5000 power traces, while Team B is able to do so with only 3000 power traces, then Team A will be listed in 2nd place, and Team B will be listed in 1st. The ranking will not impact the student's grade for that lab, and therefore displaying the ranking does not violate any privacy regulations. Rather than competing against each other, some labs may include a race against the clock, where students will need to complete various tasks / reach particular milestones within a set time. A backstory may be provided for some labs, putting the student in the position of an agent working to decode an important message or similar. We plan to add the same game mechanics to the labs in each gamified semester to ensure consistency.

## IV. Preliminary Results

To date, the course has been offered twice, in Fall 2021 and again in Fall 2022. The first time it was offered, we considered this a trial run of the course content. This initial offering was further complicated by the fact that it was taught in a hybrid fashion, synchronously online and in-person due to the COVID-19 pandemic; our approach to this hyflex laboratory course was previously documented in the literature [2]. Hence, the first "A" semester was in Fall 2022. To date, we have collected preliminary survey information from 42 students taking the course. Student demographic information can be summarized as follows:

- The student population is comprised of 62% seniors, 29% masters, 5% juniors, and 5% PhD students.
- The majority of undergraduate and masters students are pursuing a degree in computer engineering (62%). Most other students are pursuing a degree in computer science (26%). The remaining masters students are studying electrical engineering.
- About 80% of the students identify as male, and 20% as female.
- GPAs are fairly evenly distributed, with about 22% reporting a GPA between 3.75-4.00, 3.50-3.74, 3.25-3.49, and 3.00-3.24. About 10% of students report a GPA of below 3.00.

For the hardware skills inventory, we have the following initial observations:

- Most students listed familiarity with computer architecture and computer logic design (about 85%). Both of these are required courses for undergraduates.
- 1 in 3 students reported familiarity with hardware security and FPGAs. There is a FPGA elective which some students may have taken. Students who have taken computer system design would also be familiar with FPGAs. At least some students would have been familiar with FPGA basics from their logic design course, though this is inconsistent and depends on the instructor.
- 1 in 4 students were familiar with cryptography, while 20% were familiar with electronic circuits and transistors. There is a cryptographic hardware elective which some students may have taken prior to this. However, we would have expected more students to be familiar with electronic circuits, as this is also a required course for computer engineering students.
- Very few students (15% or less) reported being familiar with network / information security topics (besides cryptography, which was listed separately), software security

(12%), software or hardware reverse engineering (10% and 5%, respectively), or side channel analysis (10%). There is a software security / secure coding course available, but given the relatively low number of computer science students in the course, many may not have taken it before. The remaining topics are not offered in other elective courses.

Self-assessment data from the hardware skills inventory is summarized in Figure 2. Programming languages like C/C++ and Java, and more recently Python, are covered in core classes. Some students learn Verilog/VHDL if they have taken the FPGA elective, or else once they take the computer system design course, often in their senior year. Students are increasingly using Git in core programming classes. It is therefore expected that these are among the topics with which students felt they were very comfortable. Certain other topics and tools are generally not covered (e.g. GPGPU, PCB design, 3D modeling, etc.), and hence we do not anticipate these to increase. Since Python, C, and Verilog are utilized in the course, we anticipate these will increase. We also expect to see an increase in hardware tools such as multimeter, power supply, FPGA, function generator, and oscilloscope, since these tools are used in the course. More survey data is anticipated in the near future.

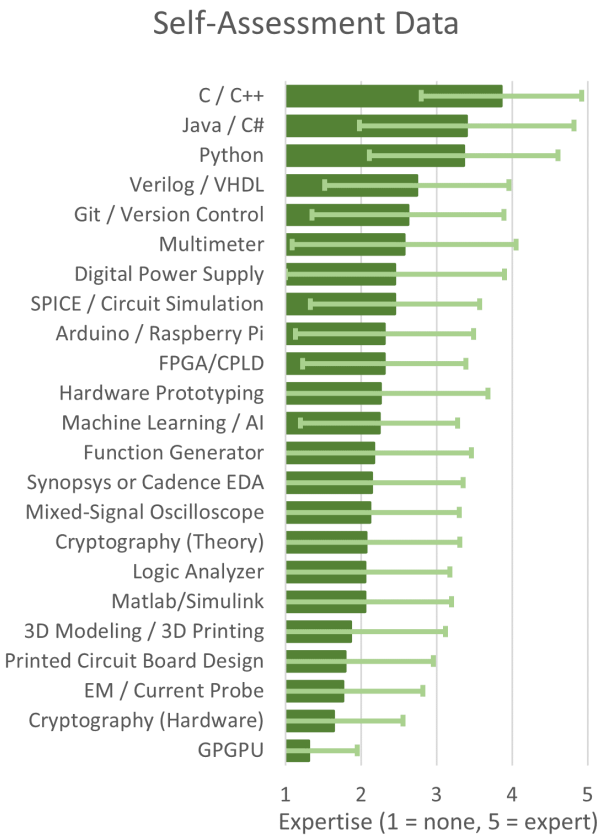


Fig. 2. Mean and standard deviation for the hardware skills inventory.

## V. Conclusion

We presented an overview of an ongoing education project that implements an innovative hands-on classroom course to teach hardware security concepts. The innovation lies in *gamifying* the student projects with the hypothesis that such gamification will improve the student learning due to its engaging nature. In order to test the hypothesis, the course is taught in two ways, namely, a traditional way and the gamified way. The course will be taught several times in an ABAB manner with gamified labs in "B" semesters. In Fall 2022 (at the time of writing this paper), the course is being taught in the traditional way. Data collection consists of analyzing student performance and survey responses. The initial self-assessment survey has some interesting findings such as most of the class considers themselves as proficient in high-level languages, but far less so in hardware topics like FPGA and embedded systems. The project emphasizes the use of low-cost open-source hardware and software as it makes the course more easily reproducible at other institutions. One of the major challenges faced so far was to prepare a VM image that runs diverse open source software packages in a cohesive and stable manner. To the best of our knowledge, this is the first time gamification has been used as an intervention to improve student learning in hardware security education.

## References


[1] D. D. Burkey, M. D. D. Anastasio, and A. Suresh, "Improving student attitudes toward the capstone laboratory course using gamification," *Age*, vol. 23, p. 1, 2013.

[2] R. A. Karam, S. Katkoori, and M. M. Kermani, "Work-in-progress: Hyflex hands-on hardware security education during covid-19," in *2022 IEEE World Engineering Education Conference (EDUNINE)*. IEEE, 2022, pp. 1–4.

[3] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work?-a literature review of empirical studies on gamification." in *HICSS*, vol. 14, no. 2014, 2014, pp. 3025–3034.

[4] S. de Sousa Borges, V. H. Durelli, H. M. Reis, and S. Isotani, "A systematic mapping on gamification applied to education," in *Proceedings of the 29th annual ACM symposium on applied computing*. ACM, 2014, pp. 216–222.

[5] D. Dicheva, C. Dichev, G. Agre, and G. Angelova, "Gamification in education: A systematic mapping study," *Educational Technology & Society*, vol. 18, no. 3, pp. 75–88, 2015.

[6] F. Panagiotis, M. Theodoros, R. Leinfellner, and R. Yasmine, "Climbing up the leaderboard: An empirical study of applying gamification techniques to a computer programming class," *Electronic Journal of e-learning*, vol. 14, no. 2, pp. 94–110, 2016.

[7] R. J. Baxter, D. K. Holderness Jr, and D. A. Wood, "Applying basic gamification techniques to it compliance training: Evidence from the lab and field," *Journal of information systems*, vol. 30, no. 3, pp. 119–133, 2015.

[8] G. Fink, D. Best, D. Manz, V. Popovsky, and B. Endicott-Popovsky, "Gamification for measuring cyber security situational awareness," in *International Conference on Augmented Cognition*. Springer, 2013, pp. 656–665.

[9] K. Fleischman and E. Ariel, "Gamification in science education: Gamifying learning of microscopic processes in the laboratory," *Contemporary Educational Technology*, vol. 7, no. 2, pp. 138–159, 2016.

[10] K. Drace, "Gamification of the laboratory experience to encourage student engagement," *Journal of Microbiology & Biology Education: JMBE*, vol. 14, no. 2, p. 273, 2013.

[11] M. T. Bonde, G. Makransky, J. Wandall, M. V. Larsen, M. Morsing, H. Jarmer, and M. O. Sommer, "Improving biotech education through gamified laboratory simulations," *Nature biotechnology*, vol. 32, no. 7, p. 694, 2014.