Benchmarking at the Frontier of Hardware Security: Lessons from Logic Locking

Coordinators: Benjamin Tan*, Ramesh Karri*,
Blue teams: Nimisha Limaye*, Abhrajit Sengupta*, Ozgur Sinanoglu† (combinational locking),
Md Moshiur Rahman‡, Swarup Bhunia‡ (sequential locking),
Red teams: Danielle Duvalsaint§, R.D. (Shawn) Blanton§, Amin Rezaei¶, Yuanqi Shen¶, Hai Zhou¶, Leon Li‖,
Alex Orailoglu‖, Zhaokun Han**, Austin Benedetti**, Luciano Brignone**, Muhammad Yasin**,
Jeyavijayan Rajendran**, Michael Zuzak††, Ankur Srivastava††, Ujjwal Guin‡‡, Chandan Karfa[x], Kanad Basu[xi]
Judges, Evaluators, Community-at-large: Vivek V. Menon[xii], Matthew French[xii], Peilin Song[xiii],
Franco Stellari[xiii], Gi-Joon Nam[xiii], Peter Gadfort[xiv], Alric Althoff[xv], Joseph Tostenrude[xvi], Saverio Fazzari[xvii],
Eric Breckenfeld[xvii], Ken Plaks[xvii]
*New York University, †New York University Abu Dhabi, ‡University of Florida, §Carnegie Mellon University,
¶Northwestern University, ‖University of California San Diego, **Texas A&M University,
††University of Maryland College Park, ‡‡Auburn University, [x]Indian Institute of Technology Guwahati,
[xi]University of Texas at Dallas, [xii]Information Sciences Institute – University of Southern California,
[xiii]IBM T. J. Watson Research Center, [xiv]CCDC Army Research Laboratory, [xv]Tortuga Logic, [xvi]Boeing, [xvii]DARPA

***Abstract*—Integrated circuits (ICs) are the foundation of all computing systems. They comprise high-value hardware intellectual property (IP) that are at risk of piracy, reverse-engineering, and modifications while making their way through the geographically-distributed IC supply chain. On the frontier of hardware security are various design-for-trust techniques that claim to protect designs from untrusted entities across the design flow. Logic locking is one technique that promises protection from the gamut of threats in IC manufacturing. In this work, we perform a critical review of logic locking techniques in the literature, and expose several shortcomings. Taking inspiration from other cybersecurity competitions, we devise a community-led benchmarking exercise to address the evaluation deficiencies. In reflecting on this process, we shed new light on deficiencies in evaluation of logic locking and reveal important future directions. The lessons learned can guide future endeavors in other areas of hardware security.**

I. Introduction

A. Hardware Security: Mitigating Threats to the Supply Chain

Integrated circuits (ICs) are the foundation of all computing systems, and the need for security spans the entire system life-cycle, from conception, through hardware design and manufacture, and across the system's operating lifetime. Hardware Intellectual Property (IP) protection is crucial, especially as the market for domain-specific computer hardware becomes more lucrative [1]. Given the high-value effort involved in hardware design, we want to prevent IP theft, counterfeiting, malicious modifications, and reverse-engineering. Effective security techniques for use in the IC supply chain are of the utmost importance for mitigating such threats.

The IC supply chain is geographically distributed and involves potentially untrusted parties that expose IP to piracy threats [18], [19]. Potential adversaries in the design flow include System-on-Chip (SoC) integrators, foundries, test facilities, and end-users, as shown in Fig. 1. Throughout the design flow, there are opportunities for overbuilding, reverse-engineering, and hardware Trojan insertion [20]. Researchers have proposed various digital circuit "Design-for-Trust" techniques, including watermarking [3], camouflaging [4]–[6], split manufacturing [7], [8], passive metering [9], [10], and logic locking [11], [17], [21]–[24], as shown in Table I.

TABLE I
Comparison of selected Design-for-Trust techniques [2]. ✓ means a technique can prevent piracy by the untrusted entity.

Design-for-Trust Technique	Untrusted Entity			
	SoC Integrator	Foundry	Test Facility	End User
Watermarking [3]	✗	✗	✗	✗
Camouflaging [4]–[6]	✗	✗	✗	✓
Split manufacturing [7], [8]	✗	✓	✗	✗
Metering (passive) [9], [10]	✗	✗	✓	✓
Logic locking [11]–[17]	✓	✓	✓	✓

We focus in this work on *logic locking* as a promising and versatile approach for protecting IP against various threats in the supply chain; it has gained traction in the wider hardware design academic community (reflected in the trajectory of publications in Fig. 2). Logic locking is an active, decade-old research area [25], with numerous proposals (e.g., [11], [17], [21]–[24]) and subsequent attacks (e.g., [13], [26]–[31]). Given logic locking's status as a frontier of hardware security, this paper contributes: (i) a critique on logic locking evaluation in the literature, (ii) community-driven benchmarking to provide a "snapshot" of the current field, and (iii) broader lessons for hardware security benchmarking. This work systematizes an important area of IC supply chain security, and as other hardware security techniques approach maturity, these experiences offer useful insights for future benchmarking efforts.

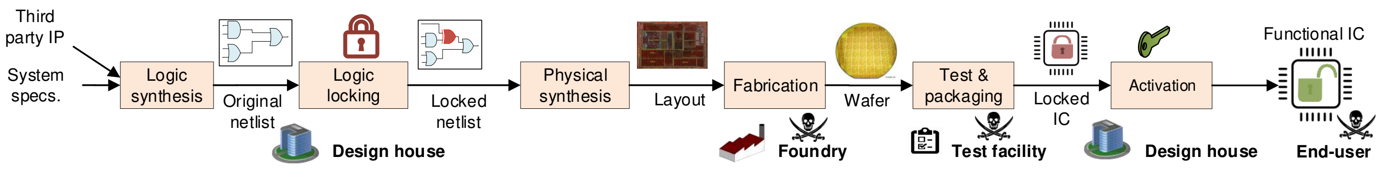


Fig. 1. The IC design flow: this involves a geographically distributed supply chain with numerous untrusted parties.

B. *What is Logic Locking?*

Broadly, IP usurpers want to take a design and use it illegitimately. As illustrated in Fig. 1, digital designs undergo a series of processes. Within a *design house*, designers integrate various IPs to satisfy a system specification; this design undergoes logic synthesis to produce a netlist. The netlist is transformed through physical synthesis to produce a layout that can be fabricated by a *foundry*, producing the IC. After manufacture, the ICs are sent to a *test facility* for testing and packaging, before being returned to the design house. Logic locking protects a design by inserting additional logic into it; the added logic "locks" the design such that a locked circuit, without the correct key, behaves erroneously. To protect combinational circuits, logic locking transforms the output into a function of the primary and the key inputs. To protect sequential circuits, logic locking inserts states and state transitions, expanding the reachable state space to include non-functional states. The netlist is locked in the *design house* before being sent to the other parties in the design flow; the locked IC is *activated* by the *design house* before distribution to *end-users*. Logic locking aims to mitigate several threats:

- An untrusted foundry or an end-user may **reverse engineer and pirate IP** by extracting a netlist from the physical IC or layout. Without the key, an attacker cannot produce a design that is functionally equivalent to the original.
- A malicious foundry may **overbuild** surplus ICs and sell them in grey markets at lower prices. However, without the secret key, the foundry cannot unlock the extra ICs.
- **Hardware Trojans** are malicious alterations to a design. A logic locked IC can prevent insertion of Trojans. The key gates alter the transition probabilities of the signals in a circuit in a manner unknown to the attacker, making it harder for them to identify stealthy locations for Trojans.
- Logic locking can hamper **counterfeiting** and cloning, as they rely on reverse engineering [19].

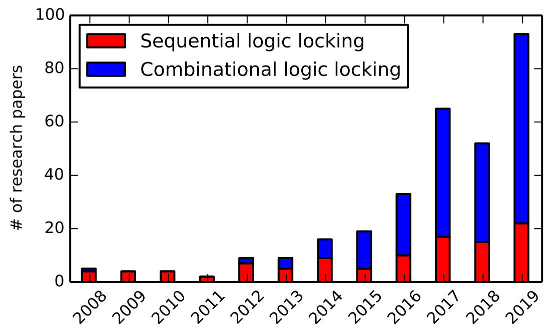


Fig. 2. Publications in logic locking over the years.

C. *What is the current landscape in logic locking research?*

While logic locking is touted by academic proponents as a promising defense against IP piracy, reverse-engineering, and hardware Trojan insertion [32], the community at-large continues to contribute numerous back-and-forth exchanges, extolling the latest triumphs against attacks or proclaiming the "end of logic locking" [33], [34]. Throughout the recent, formative years of logic locking research, various independent research groups have investigated the effectiveness of attacks and defenses alike. This raises a natural question: *What is the current landscape in logic locking research*? To answer this question, we systematize the literature and identify several shortcomings in how logic locking techniques have been evaluated. As we will explain in Section II, these include:

- Reliance on "best-effort" re-implementation of techniques
- Inconsistent locking parameters for comparisons
- Variations in comparison criteria (e.g., allowed attack time)
- Inconsistent benchmark selection
- Limited benchmark complexity
- Disagreement over "in-scope" security guarantees

Motivated to overcome these deficiencies, we sought inspiration from cybersecurity contests across software and hardware domains [35]–[41] as a means to synthesize community insights into hardware security. We highlight examples of such contests in Table II, where the exercises brought together the community to explore new approaches, benchmark techniques, and systematize knowledge. Besides synthesizing insights from the literature, we embarked on the first community-wide benchmarking of logic locking. This took the form of a red-team/blue-team "competition" with a common framework for evaluating the locking techniques and the efficacy of attacks on logic locking. By bringing together leading research groups with interested experts from government agencies and the private sector, this effort aimed to build a capability towards something akin to a NIST-style contest (e.g., post-quantum cryptography process [35]).

Contributions. From the combined efforts of the community, this study offers new insights for evaluating hardware security techniques. The contributions include:

- A critical analysis of the evaluation of attacks and defenses in the literature.
- Community-guided insights into threat models and attacker capabilities addressed by logic locking.
- Results from red-team/blue-team benchmarking. For the first time, red teams could launch attacks on a common set of benchmarks, allowing a fairer comparison of their efficacy and a measure of the resilience of the blue teams' defenses.
- Insights and reflection on the overall benchmarking process, including open questions and opportunities for investigation.

TABLE II
SELECTED COMMUNITY COMPETITIONS IN CYBERSECURITY-RELATED DOMAINS. FOR "OPEN SOURCE": ○ INDICATES THAT THE CONTEST INVOLVED/EXPECTED ATTACKS AND DEFENSES TO BE OPEN SOURCED, ● INDICATES THAT THERE WAS NO EXPECTATION FOR EITHER ATTACK OR DEFENSE TO BE OPEN, ◑ INDICATES THAT THE ATTACK ARTIFACTS WERE OPEN-SOURCE BUT NOT THE DEFENSE

Competition (First year)	Organizer	Open Source	Domain	Aims	Participants	Contributions
Crypto standards (e.g., AES, PQC) (1997)	NIST	○	Crypto	Standardize crypto algorithms	$\geq$ 300	Participants submit candidates for evaluation. Community contributes to minimum requirements, submission requirements, evaluation [35] using open comment threads and workshops
CTF (1996)	Academia+ Govt.	●	Software	Vulnerability discovery and exploitation	$\geq$ 10,000 annually	Contestants identify and exploit system vulnerabilities. These contests are offered as professional development opportunities, community-building exercises, and product security evaluation [36]
Rode0day (2018)	Academia+ Govt.	◑	Software	Automated Binary Bug Detection	$\sim$ 28 teams	Monthly "buggy binary" released. Participants aim to discover vulnerabilities. Bug injection is automated using an open-source tool [37]
Cyber Grand Challenge (2014)	DARPA	○	Software	Autonomous software vulnerability detection and patching	$\sim$ 200	Participants develop and test autonomous systems to identify vulnerabilities and develop patches for unknown software [38]. The organizers prepared the challenge sets with vulnerabilities for the competitors' automated tools to analyze and repair. Tools generated proof of vulnerabilities and deployed them against competing tools.
CSAW-ESC (2008)	Academia	○	Hardware	Hardware security	$\geq$ 300	Contest spanning numerous domains, including hardware Trojan design, detection [39], yielding the hardware Trojan corpus [42]
Hack@DAC (2017)	Academia+ Industry	◑	Hardware	Hardware Bug Detection (CTF-style)	$\geq$ 150	Participants identify bugs in an SoC. They score points by submitting bug reports identifying the root cause, a test for confirming the issue, severity assessment, and exploit code. They may submit tools for automatic bug detection. Offers insights into the challenges for manual/automated bug detection in hardware flows [40]
DPA Contest (2008)	Academia	○	Hardware/ Crypto	Side-channel resistance	$\geq$ 30	Compare side-channel techniques (attacks, acquisition techniques, counter-measures). Participants develop attacks and defenses, organizers prepare benchmarks and draw insights into sample acquisition challenges [41]
This work (2019)	Academia	●	Hardware	Logic Locking Evaluation	$\sim$ 40	Evaluate state-of-the-art combinational and sequential logic locking against numerous attacks to draw insights from participants' experiences

D. Paper Organization

In Section II, we outline the broad aims of logic locking and present our critical analysis of the logic locking literature. In Section III, we address recently proposed *invasive* attacks on logic locking. Following this, we discuss our benchmarking effort, discussing the threat model and assumptions on attacker capabilities in Section IV. In Section V, we provide brief technical background on the combinational and sequential locking techniques and then present the findings from the benchmarking effort in Section VI, where we outline the various attack approaches and the achieved attack results. In Section VII, we reflect on our efforts, examining lessons learned. Finally, we present our perspectives on the future of logic locking and its evaluation, concluding in Section VIII.

II. Critical Analysis of the Literature

A. Background: Motivations for Logic Locking

Motivation for logic locking was first presented in [11], [21], where the primary issue is that of integrated circuit (IC) or intellectual property (IP) piracy. In this scenario, a malicious party steals designs so that they can produce or distribute them for their own benefit. As such, the main objective of [11] is to discourage "unauthorized excess production and stolen masks" by making it difficult to reverse-engineer or modify a functional IC. This motivation has since inspired the research community, giving rise to threat models for both combinational and sequential circuit locking research. The intended users of logic locking include semiconductor IP designers and system integrators who want to protect high-value designs from theft and manipulation by untrusted foundries and from the threat of reverse-engineering.

B. Logic Locking Evaluation in the Literature

Table III presents an overview of work in combinational locking, highlighting techniques and settings used by authors in evaluating their locking techniques or attack approaches. We invite interested readers to examine [25] or [50] for a thorough survey of the history of logic locking and related fields. While we do not present sequential approaches here, readers can observe similar ad hoc and disparate evaluations in the literature.

In the initial proposals for logic locking, starting with Random Logic Locking (RLL), as part of EPIC [21], techniques are evaluated on generally informal, qualitative criteria, based on novel aspects, such as number of test patterns for sensitizing primary outputs to key inputs [13]. The formulation of a boolean Satisfiability (SAT)-based attack [26] provided a watershed moment—not only did they present an attack that was able to "defeat" existing logic locking techniques, they also provided a freely usable attack/defense implementation.

Spurred by the SAT-based attack, new SAT- resistant techniques have proliferated, each applied to varying subsets of ISCAS '85 circuits [51] and circuits sourced from other benchmarks (such as MCNC [52], OpenSPARC [53], and ITC '99 [54]). Different works chose different subsets (often with overlaps), but with different re-implementations of prior techniques, with different settings. For example, the choice of "how much locking" when preparing benchmarks for evaluating techniques varied across works; in some papers, e.g., [15], [46], [47] choose a 5% area overhead for guiding locking evaluation—others, such as [16], simply picked a key size (such as 128 bits). Following the dissemination of [26], the number of key bits that were successfully retrieved (in a given time) became frequent evaluation criteria for defenses, result-

TABLE III
Non-exhaustive overview of defense/attack self-evaluation: Circuits Used, Techniques (Re-)Implemented, and Evaluation Design

	Year	Technique	Evaluation Circuits (number in chosen subset)	(Re-)implemented Locking/ Attacks	Notes on Evaluation Design (e.g., key sizes)
Combinational Locking Techniques	2008	RLL [21]	ISCAS85 (2)	– / –	Informal/Qualitative security analysis focusing on the key distribution aspects
	2012	FLL [12]	ISCAS85 (10)	– / –	Informal/Qualitative security analysis
	2012	SLL [13]	ISCAS85 (9)	RLL	Brute-force analysis, key gate-type analysis, # of test pattern analysis, informal analysis
	2016	SARLock [14]	ISCAS85 (4), OpenSPARC (7)	RLL, SLL / SAT	Start of the "post-SAT" techniques, applies SAT from [26], **10–64** key-bits
	2016	Anti-SAT [15]	ISCAS85 (3), MCNC (3)	SLL / SAT	SAT from [26] with 10 hr timeout, **43–364** key bits, SLL based on **5%** area overhead
	2017	TTLock [23]	ISCAS89 (5)	– / SAT, Sensitization	**16–18** key bits, used SAT [26], 48 hr timeout
	2017	Cyclic [43]	ISCAS85 (9), MCNC (7)	RLL (for area comparison only) / –	**72** key-bits to compare overhead, no empirical attack evaluation
	2017	SFLL-HD/ SFLL-flex [16]	ISCAS89 (3), ITC99 (7), ARM Cortex-M0	FLL / SAT, Anti-SAT, 2-DIP, Sensitization	48 hour timeout for SAT, **11–14** key-bits for initial evaluation, **128** key-bits of SFLL and **128** key-bits of FLL for case study on ARM Cortex-M0
	2018	SFLL-fault [44]	ITC99 (6)	– / SAT	**128** key-bits, 48 hour timeout. Claims the same as SFLL-flex
	2019	SFLL-rem [17]	ITC99 (6), DARPA CEP SoC, ARM Cortex-M0	– / SAT, FALL	**80** key-bits, evaluated using open-source implementations from [26] and [45], **524** key-bits for SoC case study
Attacks on Combinational Locking	2012	Sensitization [13]	ISCAS85 (10)	RLL, FLL / –	Brute-force analysis, key gate-type analysis, # of test pattern analysis
	2015	SAT [26]	ISCAS85 (11), MCNC (12)	RLL, FLL / Sensitization	Released a SAT-based attack tool and tool for applying RLL, FLL, and SLL, set 10 hour timeout. Locking based on **10–50%** overhead.
	2017	SPS [46]	ISCAS85 (8) and OpenSPARC (7)	FLL, Anti-SAT / SAT	Compares SAT attack to SPS, **64 & 256** Anti-SAT key-bits + FLL based on **5%** area overhead
	2017	AGR [47]	ISCAS85 (10), ISCAS99 (2) OpenSPARC (7), MCNC (3)	FLL, Anti-SAT, SARLock / SPS, AppSAT	**512** Anti-SAT key-bits + FLL based on **5%** overhead
	2017	Bypass [48]	ISCAS85 (6), MCNC (1), EPFL (1)	RLL, SARLock, Anti-SAT	Informal comparison of attack with SAT and Removal on different techniques, mentions SLL but implements RLL for SARLock+SLL
	2017	AppSAT [27]	ISCAS85 (20), MCNC (6)	Anti-SAT, RLL / –	**16–30** Anti-SAT key-bits + **30–80** RLL key-bits
	2017	2-DIP [29]	MCNC (11)	SARLock, [32] / SAT	**8–256** SARLock key-bits with **27–1682** [32] key-bits, representing **5–25%** area overhead
	2017	CycSAT [30]	ISCAS85 (9), MCNC (12)	[32] using [26], Cyclic / SAT	**71–90** key-bits, uses [32] based on 10% area overhead, **53–837** key-bits total; illustrates shortcoming of [26], [43] as a result of specific SAT-attack implementation
	2019	Redundancy [31]	ISCAS85 (10)	RLL, Sensitization / –	Oracle-less individual and pair-wise analysis based on **5%** area overhead
	2019	Unit Function [49]	ISCAS85 (4), ITC99 (6)	RLL / –	Oracle-less self-referencing search of equivalent unit functions for **128** key bits
	2019	FALL [45]	ISCAS85 (9), MCNC (11)	TTLock, SFLL-HD / –	SFLL-HD^0, SFLL-HD^h with **10–128** key-bits, 1000 sec. timeout

ing in various time limits for running attacks, ranging from seconds (set in [45]) to days (set in [44]). From examining the variability in evaluation design, several questions arise about the present state of logic locking attack/defense evaluation:

- What is the right key size?
- Are oft-used benchmarks (like ISCAS) representative of "typical" IPs that logic locking might protect?
- What is a reasonable attack timeout?
- What are "realistic" capabilities of attackers? (we examine this more specifically in Section III)

The answers to these questions differ across studies, which brings to question the generalizability of claims about whether locking is "provably secure" [16], will "most likely never succeed" [33], or somewhere in-between. We take this uncertainty as an opportunity for a coordinated evaluation.

What is the right key size? Various papers set the "amount" of locking using some holistic measure—usually as part of some security/power/performance/area trade-off (e.g., locking based on 5% overhead as in [47]). In other works, key size is often set independently of the target circuit characteristics, (e.g., in [45], where key size is set to a maximum 128 bits). When key size is set by researchers for their attack evaluations, it is hard to know if the re-implementation settings are aligned with defense designers' intentions. Furthermore, while there may be some overlap between the circuits used for evaluation, there is not always a 1:1 correspondence, which makes comparisons somewhat inconclusive.

Are ISCAS '85 benchmarks enough? The most-used benchmark circuits come from the ISCAS '85 collection [51], featuring circuits from about 6 to 3000 gates with dozens to hundreds of inputs and outputs. Even where ISCAS '85 circuits are used, the selection of circuits within the benchmark set varies from work to work. In addition, these designs are small compared to "realistic" designs, such as the IPs used in SoCs (e.g., in [17], the IPs are around 16∼156K gates, orders of magnitude larger than the ISCAS '85 circuits). As the ISCAS '85 circuits are not necessarily representative of the scale of complex modern IPs that will benefit from logic locking techniques, whether the claims about the scalability of locking (and corresponding attacks) hold for "realistic" scenarios requires further examination. While not yet widespread, recent works [17], [55] have started using realistic SoCs, such as the Common Evaluation Platform [56] which has a UCB Rocket-Chip and integrated DES3, GPS, MD5, RSA, SHA256, DFT, IDFT, and IIR cores connected by a AXI4-Lite bus.

What is a reasonable attack time out? Similar to the variability in key size and circuit selection, quantifying attack resilience/efficacy also varies from study to study. The community is evaluating security by determining what is "practical", at least empirically. In some locking technique studies, authors aim for resilience to the SAT-based attack [26] (e.g., [15], [23]) as measured by whether the SAT-based attack times out by taking longer than a pre-specified bound. As attack time is influenced by many factors such as the circuit complexity and key sizes and not determined solely by the attack algorithms, another criterion may be complexity in terms of the number of iterations needed to recover a key. There are no guarantees that the underlying execution platform resources are consistent across evaluations, and no contextualization/justification as to what constitutes a "reasonable" amount of time that a dedicated attacker would be willing to devote before an attack becomes uneconomical. Another missing piece of the puzzle is whether "attack time" should consider the wider effort required of an attacker; for example, should the time taken to setup tools, understand the benchmark formats, and so forth, be a factor in a holistic evaluation?

C. Shortcomings in Evaluations: A Critical View

As evident in recent work, there is a lack of agreement with respect to metrics for the trade-off between security and overhead [55], [57]. As such, there are shortcomings in the framing, analysis, and evaluation of attacks and defenses, including:

Relying on "best-effort" re-implementation of techniques. Evaluation of attack and defense techniques are somewhat ad hoc in the literature; independent groups typically select what they find to be the state-of-the-art attack/defense techniques, applying them on relatively small scale designs (either bespoke, or drawn from benchmarks designed for other purposes). Due to a lack of coordination or implementation availability, the techniques are often re-implemented from (often incomplete) descriptions found in literature.

This reduces transparency of the evaluation process. While techniques are somewhat scrutinized (and there is a case to be made on the merits of replicability), there are potential biases at play (for example, choice of circuits and key sizes) and limitations on the quality of the (re-)implementations used to compare techniques. Furthermore, only some logic locking attack and defense implementations are open-sourced. To improve attacks and defenses, attackers need feedback on how well their techniques break locking, and defenders need feedback on how well their techniques perform under scrutiny.

Inconsistency in setting parameters for comparisons. In addition to the re-implementation of techniques, attack and defense techniques feature several parameters. These include which combination of techniques to apply, the number of key bits, which attack techniques to compare with, and attack time-out limits. Given the wide variety of settings, direct comparisons between different techniques are difficult to make, so conclusions related to relative performance/overhead impacts vs. attack success, etc. do not necessarily provide clear guidance to practitioners as to appropriate key sizes or security/overhead trade-offs. Moreover, different attacks are performed with varying computational resources, from student desktop-class machines through to university high-performance clusters, further complicating direct comparisons.

Variations in comparison criteria. When comparing locking and attacking techniques, researchers use numerous criteria to compare their works to prior art. These include area overhead, performance impact, and resilience against a chosen attack (e.g., measured as time taken to perform a SAT-based attack [26], or a specific *implementation* of a SAT-based attack). On this front, individual contributions to the literature are somewhat scattershot: some works might trade-off area overhead against number of SAT-based iterations, whereas others might use analytic measures (e.g., theoretical corruptibility). Different groups choose different evaluation criteria, while making broader claims about scalability, attack resiliency, or attack effectiveness against types of defenses. As observed in [55], there are no unified metrics in terms of circuit size, SAT-attack execution time, or technology libraries.

Limited benchmarks. As mentioned earlier, numerous works use a limited set of evaluation circuits, typically drawn from the ISCAS '85 benchmarks [51], ITC '99 set [54] and the MCNC (ACM/SIGDA) benchmark suite [52]. While they may not match the complexities of IPs in modern systems-on-chip (SoCs), they provide valuable insight into how locking techniques change logical structure and affect overhead. Nevertheless, they may miss important issues/flaws that can arise when applying logic locking to large modern designs. While these oft-used benchmarks provide some commonality across works in the literature, they provide limited range in terms of reflecting different applications and use cases, especially as the community moves forward towards developing more useful and robust techniques.

Furthermore, with respect to emulating attack scenarios, the use of known benchmarks provides advantages to an attacker (beyond the scope of the threat model). Additionally, ISCAS and ITC benchmarks were developed by the IC test community for evaluating automatic test pattern generation (ATPG) coverage. Other research communities, such as those working on hardware security, have adopted these benchmarks. Crucially, for logic locking, these benchmarks do not have verification testbenches to prove that locked circuits behave identically to original circuits or that a key has correctly been applied. Another practical issue is that these benchmarks are often in `BENCH` format (from the ISCAS benchmarks [51]), which is not easily compatible with several EDA tools.

Limited resources for coordinating. Aside from the effort we present in this paper, there have not been many opportunities for a red team/blue team style evaluation of attacks and defenses, where an impartial party facilitates the interaction between participants and allows organized sharing of resources (e.g., benchmark circuits and external judging). Without this coordination, the barrier of entry into the logic locking attack/defense domain can be prohibitively high, as teams need to prepare both attack techniques and evaluation artifacts. Running an exercise that frees teams from the need to re-implement allows attackers and defenders to focus on their techniques; this also enables a controlled and fair comparison across techniques (and their implementations).

Disagreement over "in-scope" security guarantees. There are myriad threat models that aim to be addressed by myriad defense techniques. In the existing literature, defender goals (when proposing a locking mechanism) and attacker points-of-view (in terms of what constitutes the "breaking" of a technique) do not always align. For example, the work in [17] assumes an attacker that can reverse-engineer a netlist from a physical layout (e.g., GDSII files) and also acquire a working chip (e.g., from the "open-market"), setting aside invasive attacks as orthogonal (and addressed by advances in physical, package-level countermeasures). In contrast, recent critiques of logic locking [33], [34] propose invasive attacks, such as optical probing. Such attacks do not identify weaknesses in the fundamental strength of the proposed locking techniques, but instead target the underlying implementation technologies—these vulnerabilities can be addressed with suitable technology-based solutions that build on top of logic locking in a defense-in-depth approach [58]. We discuss more challenges to invasive attacks in Section III.

D. Informing our Benchmarking Exercise

Given the plethora of evaluation settings, we were motivated to bring the community for a coordinated benchmarking effort. In other words, the aim of this benchmarking exercise was to provide a level playing field for comparing different attacks against locking techniques, allowing teams to field their "best efforts" by studying a common set of circuits within a common time-frame. Our benchmarking exercise was thus designed to address these shortcomings in the current landscape by:

- agreeing on a threat model for the benchmarking exercise (see Section IV);
- setting common goals to compare attack techniques (i.e., key recovery, see Section IV);
- preparing a shared benchmark suite with clearly defined locking parameters (see Section V);
- focusing on larger, realistic benchmarks (see Section V);
- empowering the teams to focus on their attacks/defenses, without fretting about re-implementing others' works (see Section VI); and
- coordinating the community's effort (see Section VI).

III. On Invasive Attacks and Assumptions

In [33] and [34], authors proposed new threat models based on the availability of advanced probing techniques such as e-beam [59] and laser probing [60], [61] that were developed for IC failure localization analysis. Although spontaneous light emission techniques [62], [63] were not mentioned in [33], [34], they have also successfully been used to probe internal IC nodes. These probing tools were originally invented and developed for IC failure analysis and chip characterization [60], [63]. Typically, they are used for defect localization through global inspection (using light emission maps and electro-optical frequency mapping) and then local probing of the specific nodes. Usually these techniques are time-consuming and require preliminary electrical test and diagnostics aimed to narrow down the probing areas.

These tools have been expanded and used for hardware security by developing automated data collection, image processing, and information extraction [62], [64]. Some applications include identifying functional blocks and detecting Trojans [62], and reading the contents of 14 nm SRAM [64].

Fundamentally, there are two issues when attackers want to use optical probing techniques for observing the key gate signals or key latches/flip-flops. One is the limitation of spatial resolution, and the other is identifying the probing location. As the technology evolves to 7 nm and below, it becomes very difficult for any non-invasive optical probing tool to be able to see individual gates, even using Solid Immersion Lenses (SILs) [65]. The optical spatial resolution of a state-of-the-art probing tool equipped with SIL is about 200 nm which can be calculated using the Rayleigh criterion R=0.61λ/NA, where λ is the light wavelength (e.g., 1064 nm or 1340 nm for commonly available probing lasers), and NA is the SIL numerical aperture (e.g., $\sim$3). The same principle can be applied to the resolution of transistors that emit light emissions when they are powered up or switching. These emissions are broadband and reach a peak around 1.8 m for new technology nodes [66]. Therefore, using light emission for probing also suffers of resolution issues. However, even with these limitations and challenges, light emission can be used to detect some types of Trojans [62] due to advances in image processing and analysis techniques. This is because the physical size of Trojans is large enough so that optical probing can identify additional circuitry compared to the original design.

For extracting the logic locking key, however, one needs to identify the gates used in locking—this is much smaller compared to hardware Trojans. In [34], a 28 nm FPGA was used to demonstrate that the key values can be extracted using laser probing. Although this proof-of-concept example shows the success of laser probing, it will suffer when measuring random logics manufactured in the latest technology node. The diffraction limitation fundamentally defines the resolution of optical probing, as mentioned earlier. For scaled nodes, the electro-optical spots are often a cluster of many gates and it is very hard or impossible to interpret them. It is even harder to derive the logic key values from these spots. For random logic, designers can intentionally hide the key gates using a more compact layout. From the semiconductor process side, designers can choose heavily doped silicon wafer so that the near infrared light is absorbed by the silicon. This makes it hard to see through the backside. Therefore, key bits can be hidden without using design tricks as an anti-tamper feature.

Localizing the region of the chip containing the locking logic is another challenge. In contrast to regular chip diagnostics where test engineers have some targeted test patterns and responses to guide where to probe, designers of locked circuits could make a design where keys are hard to find by probing. Although recent works [33], [34] assume that a malicious foundry can reverse engineer the IP core from the GDSII file and identify the key gates and registers, it is in fact a challenge to localize them, in particular for random logic. Together with the limits of spatial resolution at the advanced nodes, it is challenging to reveal the key bits using probing.

Fundamentally, the goal of logic locking is to protect IP during manufacture. Protecting IP after manufacture and during use is the province of anti-tamper technologies. Both technologies must be used in a coordinated fashion to protect a chip throughout its life cycle. Ideally they would be complementary approaches that make both the locking and the *anti-tamper stronger than they would be in isolation. This is a promising area for further study.*

IV. Benchmarking: Threat Model and Goals

A. Wider Attacker Motivations

Following our critical examination of the literature, we now turn our attention to benchmarking. In discussing any potential security problem, it is important to characterize potential attackers' motivations, targets, and capabilities. Rajendran *et al.* first described the potential threat posed by an attacker that can (1) reverse-engineer a locked netlist, and (2) access a functional, unlocked design (i.e., acquired from the open market) [13]. Access to a reverse-engineered netlist enables attackers to perform structural analysis of sequential [22] and combinational circuits [13]. With an unlocked netlist, attackers have an oracle, giving rise to query-based attacks, such as those that cast unlocking as a Boolean satisfiability (SAT) problem, e.g., as originally formulated by Subramanyan *et al.* [26]. This threat has oft-appeared in the literature, where it is posited that malicious designers, in a chip fabrication plant, are well-positioned to reverse-engineer the GDSII file of a design to obtain the locked netlist, and malicious end-users or IC testers are well-positioned to access a functional, unlocked chip [17]. Attackers that conspire and coordinate between these pre- and post-fabrication threat actors thus represent a plausible "worst-case" most-capable attack scenario that logic locking techniques seek to address.

B. Benchmarking Threat Model and Attacker Goals

Using this threat of colluding malicious actors in the design flow as our starting point, red and blue teams discussed various challenge goals focused on unlocking the correct functionality of a locked design, i.e., working towards functional equivalence between an unlocked design and the corresponding attacker-recovered design. Our discussions worked towards deciding what would be considered "in-scope" for our initial benchmarking effort. As evident in the literature, this aim is framed as a problem of determining the secret key (or unlocking key sequence in sequential locking), as the recovery of the key entails full recovery of a locked IC's functionality.

In combinational locking, we also considered the possibility of *approximate* functionality recovery—i.e., we discussed if a "partially" successful attack is meaningful, where an attacker only recovers a subset of the key or produces an approximately functional circuit. A "partially unlocked" circuit would produce input/output functionality that matches some, but not all, of those from the original design. The utility of a partially unlocked circuit depends on the application context: a circuit that has a low error rate, relative to the complete input/output space, could constitute a successful attack; in another case, perhaps with regards to mission critical designs, no error is tolerable at all. Alternatively, it may be the case that no error in a *user-defined* input/output subset is allowed in the partially unlocked circuit—whether or not the attacker has prior knowledge of the targeted design and its use depends on the broader context, such as the availability of collateral (e.g., datasheets, whitepapers, or patents).

The benchmarking coalesced around the essential threat that logic locking thwarts, i.e., embracing the *key-centric* model. The red teams set out to recover the secret key, with or without oracle access. As we re-iterate in Section VII, partially unlocked circuits are useful in some contexts obviating recovery of the full key.

C. Benchmarking Scope

In the wider context of electronic design reverse-engineering, real-world attackers need to perform various tasks to recover knowledge about a design, depending on the design artifacts they are able to retrieve. For example, attackers need to re-produce the netlist; given a netlist, an attacker might first need to perform a functional analysis to identify the components-of-interest in the design [67]. Given a manufactured IC, attackers first need to depackage, delayer, and carry out some form of imaging/microscopy to ascertain the circuit structure [68], after which the netlist can be recovered for further analysis. Attackers may want to gain information from probing devices and measuring side-channels (such as power or EM emissions). These techniques are by no means trivial, and represent a significant level of investment by them.

As logic locking provides one security modality among many possible complementary defenses, we focused the scope of our benchmarking effort to the evaluation of logic locking techniques themselves, setting aside the "upstream" challenges for recovering the netlist to begin with. To that end, we assumed that a correct, locked netlist is available to the attacker, and that, where we make available an oracle, barriers to acquiring an unlocked working device are non-existent.

V. Benchmarking: Overview and Preparation

A. Participants and Timeline

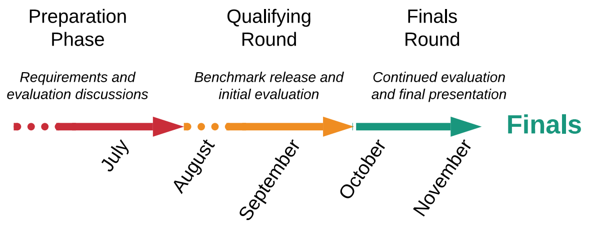

Preparation Phase
Requirements and evaluation discussions
Qualifying Round
Benchmark release and initial evaluation
Finals Round
Continued evaluation and final presentation
Finals
July
August
September
October
November

Fig. 3. Timeline of the Benchmarking Process

The benchmarking involved over 40 experts registered in 18 teams from 14 affiliations in the USA and India. The teams comprised experts in VLSI test, hardware security, and academics working at the forefront of combinational and sequential logic locking research. Additionally, we invited seven external judges from industry and government agencies alongside some industry observers. They interacted with/quizzed the participants at the finals event, where the attack teams presented their approaches and outcomes.

The benchmarking effort involved two rounds, as illustrated in Fig. 3. The qualifying round lasted $\sim$1.5 months. The red teams attacked the locked designs created by the blue teams while maintaining a live journal on their attack methods and

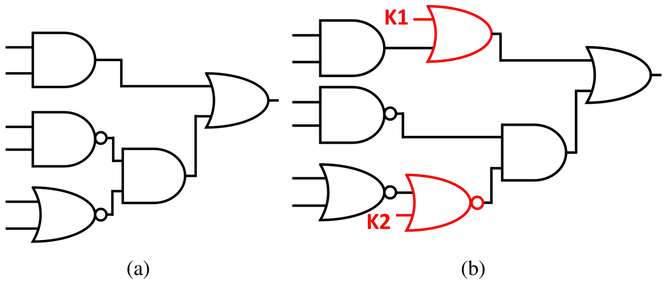


Fig. 4. (a) Original IP (b) Locked IP with two randomly inserted key gates.

the tools used. At the end of this round, the red teams submitted preliminary results (i.e., key bits and/or input sequences) for feedback. Red teams with promising approaches were down-selected as finalists. The finals round lasted ~1 month and involved further attack work, culminating in the delivery of a final report, poster, and in-person presentation to the judging panel. During the challenge, we invited teams to reach out to the blue teams as required (via the challenge coordinators) for clarification or technical support. There were no restrictions on the red teams with respect to compute resources or tools[1].

B. Benchmark Preparation

Participants focused on combinational and sequential locking separately, with two expert blue teams preparing respective sets of locked designs. Red teams had the freedom to choose which class and size of circuits to attack. This section describes the locking techniques selected for evaluation and provides details on the size/complexity of the locked designs. The details are provided from the perspective of the blue teams for combinational (*Team X*) and sequential locking (*Team Y*).

1) Combinational Locking Overview: In combinational locking, the design IP is locked with key inputs by inserting some logic gates; without the correct value provided at these key inputs, the design IP will produce incorrect outputs and fail to work. The correct key value is kept secret from untrusted entities. Fig. 4 shows an example logic locking implementation using random logic locking (RLL) [21] which adds "key gates" throughout a design to corrupt the output with incorrect key inputs. In this benchmarking, *Team X* adopts two techniques in tandem: Stripped Functionality Logic Locking (SFLL-rem) [17] and RLL [21]. The attackers need to circumvent both layers of defense to break the overall defense.

The first logic locking scheme is designed to mitigate the threat of SAT attack; details of the inner workings of SFLL-rem are available in [17]. SFLL-rem uses point-functions to increase the effort of the SAT attack by eliminating one key in each iteration out of a possible 2^N keys, where N is the number of bits in the key. Although it is SAT attack resistant, this defense is vulnerable to approximate attacks.

To thwart approximate attacks [27], [29], the benchmarks implement RLL [21] on top of SFLL-rem to increase the output corruption when an incorrect key is used (i.e., to produce incorrect functionality for a partially recovered (approximate)

[1]a tacit expectation was that the teams did not subvert the spirit of the competition by reverse-engineering the oracle (executables).

TABLE IV
COMBINATIONAL CIRCUIT BENCHMARKS. SMALL BENCHMARKS ARE LOCKED WITH A 40-BIT RLL KEY AND 40-BIT SFLL-REM KEY. MEDIUM BENCHMARKS ARE LOCKED WITH A 60-BIT RLL KEY AND 60-BIT SFLL-REM KEY. LARGE BENCHMARKS ARE LOCKED WITH 80-BIT RLL KEY AND 80-BIT SFLL-REM KEY. THE BONUS CIRCUIT WAS LOCKED WITH 128-BIT RLL KEY AND 128-BIT SFLL-REM KEY.

Competition benchmark	# Inputs	# Outputs	# Gates
small (derived from b20_C)	522	512	20226
medium (derived from b22_C)	767	757	29951
large (derived from b17_C)	1452	1445	32326
bonus (Cortex-M0)	892	1746	16164

key). RLL randomly inserts key gates into the netlist which may or may not at times provide 50% output corruption. More sophisticated techniques such as fault-analysis based logic locking (FLL) [12] and strong logic locking (SLL) [13] can be used to complement SFLL.

2) Combinational Locking Benchmark Preparation: Team X applied SFLL-rem + RLL to 7 different circuits comprising two variants per reference design (i.e., one provided with an oracle, the other without) derived from academic benchmark circuits (taken from ITC '99 benchmark suite), and a locked Cortex-M0 microprocessor (as a bonus challenge). The circuits are named small, medium, and large, based on relative gate counts and listed in Table IV. The team chose key sizes for each of the small, medium, large, and bonus circuits, according to their gate count. For the small, medium, and large circuits, we choose 40, 60, and 80 bits of security, respectively, in part to make the competition approachable. The bonus circuit, however, is locked with 128-bit security.

The combinational locking process involves three steps:

1) Remove all traces of the benchmark identity. The modifications prevent recovery of the functionality by comparing the circuit against public benchmarks[2].
   a) Change gate types of randomly selected gates.
   b) Rename internal nets by converting circuit to AND-Inverter gate (AIG) format.
   c) Rename I/O ports.
2) Lock modified circuit with SFLL-rem.
3) Lock SFLL-rem locked circuit with RLL.

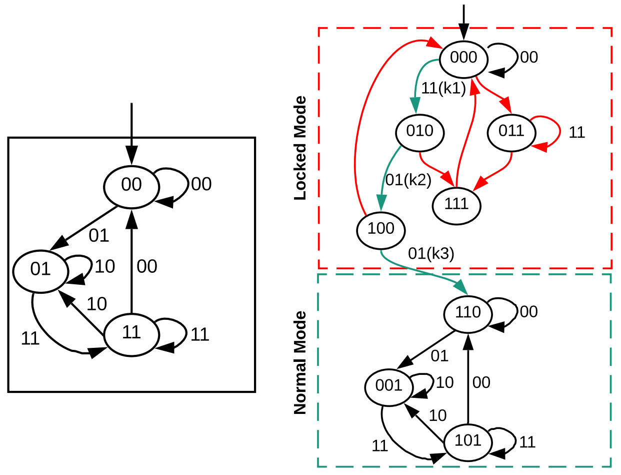
Team X verified the unlocking of the design using an open source equivalence checker [26]; it requires the locked netlist, the original netlist, and the key value. To prepare an oracle for the locked circuits, *Team X* converted the modified benchfile to executable permitting access to I/Os and not the internal circuitry. This mimics access to a working chip obtained from the market. Since the competition does not provide the original netlist, the red teams cannot use equivalence checks to verify their key bits. Hence, the red teams report back to the blue team (via the organizers) who confirm how many of the key bits recovered are correct.

3) Sequential Locking Overview: Team Y uses an improved variant of state space obfuscation [22] for sequential locking. Since the correct operation of any IP depends on the control logic (i.e., a function of inputs and control states embedded in a finite state machine (FSM) into the design) the sequential

[2]Cadence LEC can provide a patch file to recover original functionality.

TABLE V
STATISTICS OF THE SEQUENTIAL CIRCUIT BENCHMARKS

Attack Model	Competition Benchmark	Academic Benchmark	No. of Inputs	No. of Outputs	Length of Key Sequence
Warm-up	Tiny	Custom	6	71	18
With Oracle	Small	i2c	18	14	51
	Medium	md5	41	35	63
	Large	s35932	36	32	134
Without Oracle	Small	s15850	13	87	52
	Medium	s13207	11	121	103
	Large	uart	21	13	162



(a) (b)

Fig. 5. (a) Original state machine (b) Locked state machine with added state register (increasing number of reachable states from three to eight)

locking defense involves expanding the functional state space by inserting additional state registers into the FSM. The added and the existing state registers form a combined FSM which dictates the *normal* or *locked* operating mode of the design.

To enable the normal mode of operation, an authorized user needs to apply a sequence of unlocking key inputs through the design's primary inputs. This causes the extended state machine to traverse along a specific "unlocking" state sequence. Without the correct key sequence, the locked design will remain in, and transition between, a set of non-functional states, producing non-functional (i.e., corrupted) outputs. Thus, to enable the normal operation mode, an attacker would:

1) Find the inserted state registers used in locking,
2) Find the unlocking key sequence, and
3) Extract the functional states from the locked design.

The locking mechanism is illustrated in Fig. 5. To enable the normal operation, the *green path* must be traversed by applying a sequence of key inputs *K1, K2, and K3* where each of the key inputs can comprise varying subsets of the primary inputs. If an incorrect key input is applied, the *red path* is traversed and the normal mode is never reached. As the IP is in the locked mode, it remains non-functional. Unlike combinational locking, sequential locking re-uses primary inputs as key inputs. Based on the constraints and level of the security, the key width and the length of the key sequence is altered.

4) Sequential Locking Benchmark Preparation: Each design is locked using different lengths of key sequence as shown in Table V. To lock a design, extra state registers are inserted to increase the reachable state space of the design. The expanded state space overlaps with the existing state space; the state transition function of the existing state machine is modified to use the additional states available from the extra state registers, including the addition of the unlocking input sequence. The number of gates, the number of inputs/outputs and the number of state registers in the design determine the complexity of locking and difficulty of unlocking.

The locked designs are synthesized using LEDA 250nm technology node. *Team Y* renamed the wires and cells to prevent attackers from isolating the obfuscation logic by looking at the names. However, primary input/output names were kept as is. The attackers are given white-box access to the locked netlists, *i.e.*, they have access to the inputs, outputs, and individual cells of the design. For oracle-less attacks, the attackers were given the locked netlist. For oracle-based attacks, a locked netlist and the original netlist (i.e., the design before locking) were provided. To verify the results, the attacker can simulate the locked design by applying the key sequence, followed by a random pattern. The output is compared to the output of the oracle for the same input.

VI. Benchmarking Results

A. Results Overview

We present the finalists' attack results on combinational and sequential locking in Table VI and Table VII, respectively. In the combinational locking attacks, several teams recovered all RLL key bits. Each table presents the teams, their attack approach and their chosen attack scenario (oracle or oracle-less). This is followed by attack success on the various benchmarks represented by the number of key bits recovered. Red teams reported that their attacks recovered a number of key bits. The blue team checked to see how many matched up. We present the result as "x/y", i.e., the number of bits, x, verified as correct, out of the bits recovered by the attacker, y. No team was able to recover the complete set of SFLL keys.

In the attacks on sequential locking, none of the teams reported an entirely correct sequence and set of key input ports. One attack recovered a subset of the key input ports used for key assertion—in Table VII, this is represented as "a/b", where a out of the b input ports used as key inputs were correctly identified. The attack attempted to identify the key sequence; this is reported as "c/d" where c is the sequence length recovered by the attack, while d is the actual length.

B. Outline of Attack Approaches

The red teams proposed interesting attack directions, in some cases incorporating those proposed in the literature. None of the attacks were able to successfully recover the entire key. In this section, we outline the attack attempts and their success. The appendix presents more detail on the attacks.

- **ATPG-based** (Oracle-guided) *Team A* applied the sensitization attack [13] on combinational locking. While the team identified the primary inputs connected to the key inputs, they failed to recover the SFLL-rem key. *Overall,*

TABLE VI
COMBINATIONAL LOCKING ATTACK RESULTS.

Team	Approach	Attack Scenario	Benchmark							
			Small (40+40)		Medium (60+60)		Large (80+80)		Bonus (128+128)	
			RLL	SFLL	RLL	SFLL	RLL	SFLL	RLL	SFLL
A	Key Sensitization	Oracle	40/40	-	60/60	-	80/80	-	-	-
B	Hamming Distance-based Attack	Oracle	30/30	-	50/50	-	72/72	-	-	-
C	Automated Analysis + SAT	Oracle	11/18	-	31/50	-	10/34	-	-	-
D	Sub-circuit SAT	Oracle	17/17	-	29/29	-	-	-	-	-
E	Redundancy-based	Oracle-less	28/28	4/12	35/35	23/28	45/45	0/51	66/66	8/27
F	Bit-flipping Attack	Oracle	40/40	-	60/60	-	80/80	-	-	-
G	Topology guided attack	Oracle-less	15/32	-	19/50	-	36/73	-	75/108	-

TABLE VII
SEQUENTIAL LOCKING ATTACK RESULTS.

Team	Small		Medium		Large		Attack Scenario
	# input ports used for key assertion	Length of key sequence (# clock cycles)	# input ports used for key assertion	Length of key sequence (# clock cycles)	# input ports used for key assertion	Length of key sequence (# clock cycles)	
A	5/10	10/52	4/5	12/103	4/19	11/162	Oracle-free
	-/5	-/63	5/16	10/51	4/26	10/134	Oracle

they recovered 100% of the RLL key bits and none for SFLL-rem. The team also used an ATPG-based method to attack sequential locking based upon [69], but this did not correctly recover the key sequence.

- **Hamming Distance (HD)-based Attack** (Oracle-guided) *Team B* proposed a divide-and-conquer approach to attack the combinational locking. To attack SFLL-rem, they pick the protected input patterns (PIP) one at a time and apply the SAT attack. *This team was able only to recover most of the RLL key bits and none of the SFLL-rem key bits.*
- **Automated SAT** (Oracle-guided) *Team C* attempted a divide-and-conquer approach by dividing the circuit into smaller logic cones, starting with primary outputs and performing a SAT attack on the smaller cones, focusing on cones with fewer key inputs. *The team recovered fewer than 50% of RLL key bits and no SFLL-rem key bits.*
- **Sub-circuit SAT** (Oracle-guided) *Team D* proposed a SAT attack on combinational locking by simplifying the circuit using two strategies and then applying SAT attack. First, they find an output which depends on one key input. Second, they find a key input which propagates to one output only. *They recovered fewer than 50% of the correct RLL keys and none of the SFLL-rem key bits.*
- **Circuit Redundancy** (Oracle-less) *Team E* centered their attack on circuit redundancy [31] by observing that a post-synthesis insertion of key gates yields invalid design alternatives that do not adhere to well-established design characteristics, specifically the redundancy level, thus giving away the incorrect key bit values. *The team recovered more than 50% of RLL key bits, all correctly. They recovered fewer than 50% of the SFLL-rem key bits.*
- **Bit-flipping attack** (Oracle-guided) *Team F* devised an attack on the combinational locking using bit flipping [28]. *This attack recovered all RLL key bits in under a minute for the largest benchmark and none of the SFLL-rem key bits.*
- **Unit Function Search Attack** (Oracle-less) *Team G* attempted the Unit Function search attack on the combinational locking [49]. If one or more key gates are placed in an instance of repeated unit function (*UF*) during the locking of a circuit, the original netlist can be recovered by searching the equivalent unit functions (*EUFs*) with all hypothesis keys. *This attack recovered fewer than 50% of RLL key bits, and no SFLL-rem key bits.*

VII. Reflections and Lessons Learned

Following the benchmarking effort, we reflected on the undertaking and solicited feedback from participants and judges. This section describes observations and insights we drew from examining the competition submissions and continuing discussions amongst the coordinators, red-team and blue-team participants, judges, and external observers.

A. Observations and lessons for logic locking

Important Findings for Logic Locking. Table VIII summarizes the findings from the benchmarking process. SAT attacks remain a popular element of the attacks on SFLL-rem + RLL, but requires oracle access. Exploration on oracle-free approaches is an open challenge. State space obfuscation presents interesting challenges for attackers, especially without access to a scan-chain; attackers need to decide how much to unroll a design to perform analysis. Sequential locking is less familiar to the community and hence is less mature and will benefit from wider dissemination. As it stands, SFLL-rem and State Obfuscation remain "unbroken" based on this benchmarking effort, with the fielded attacks offering only partial "success" in key recovery.

On Participant Prior Experience. Most participants in the benchmarking exercise were experienced researchers in VLSI testing, hardware security, and specifically logic locking. Several teams featured members with a mix of background familiarity, where undergraduates worked with doctoral candidates that had published works on attacks/defenses in logic locking. In the qualifying round, some teams comprised students who were introduced to the concepts of logic locking in class a few

TABLE VIII
IMPORTANT FINDINGS (SAT: SATISFIABILITY SOLVER; ATPG: AUTOMATIC TEST PATTERN GENERATION).

Locking Technique	On SAT-based Attacks	On ATPG-based Attacks	On Structural Analysis	Important Takeaways
SFLL-rem + RLL (Combinational)	Remains the technique-of-choice for attacking LL. Successful in attacking RLL.	Emerging attack formulation, showing some promise even without oracle access	Crucial for attacking locked designs in a more scalable, piece-wise manner	Most attempted attacks require oracle access; the potential for oracle-free approaches require further exploration. RLL can be attacked successfully while SFLL-rem remains resilient to attack
State Space Obfuscation (Sequential)	Not attempted, requires scan access or appropriately unrolled design	Also requires unrolling, but # of cycles to unroll is exponential in number of flip-flops and initial (reset) state	Important to find the added state registers	Not yet as mature/familiar as combinational locking, perhaps requiring further dissemination for a more thorough evaluation.

weeks before the contest. Furthermore, most teams concerned themselves with combinational locking only—suggesting that sequential locking is less familiar.

On Tools Used. Participants described a number of different tools used to perform attacks on the benchmark circuits. These include various SAT solvers, ATALANTA ATPG tool [70], and a lot of scripting (typically in languages like Python or Perl). Many teams automated netlist parsing and analysis, although some teams started with a level of manual inspection.

B. Reflections on the benchmarking process

On Logistics and Communication. Throughout the benchmarking process, the coordinators acted as intermediaries between red teams and blue teams, and were keeping all people in the loop. The primary medium for communication was e-mail; participants were invited to maintain a live report to detail their efforts throughout the exercise, although these were often left untouched until the hours before the deadline. Ensuring that e-mails were received and noted is particularly challenging, although perhaps understandable given that participants may have had numerous competing concerns (i.e., as graduate students, many participants had coursework and other academic pursuits with respect to the academic calendar). Maintaining interest and incentivizing participation are difficult challenges; we tackled this by being in frequent contact with the red teams, blue teams and the judges[3].

On Benchmark Management. There were a number of challenges in terms of benchmark preparation, including how best to prepare and validate the benchmark circuits, and subsequently, how to distribute them. In our benchmarking effort, we relied on good-faith, best-effort practices from the blue teams. The blue teams independently prepared benchmarks and accompanying support material (e.g., oracles) after consultations with the community. Once the benchmarking commenced, the coordinator facilitated communication between red teams and blue teams, accommodating requests where reasonable (for example, recompilation of oracles for different execution platforms). While this approach worked well, future iterations will benefit by additional documentation and vetting of the benchmark circuit preparation process by parties external to the blue teams. Furthermore, future iterations could adopt standard formats for benchmarks (e.g., netlists in Verilog or VHDL) synthesized with an open-source technology library, for better tool interoperability.

[3]We offered a funded trip to the in-person finals for the student finalists as additional motivation.

On Result Assessment/Verification. Red teams submitted their attack outputs to the coordinator; these were then passed on to the blue teams for correctness assessment. While this process works in terms of assessing key recovery success, our benchmarking effort was not able to evaluate other metrics; while the scalability of an attack can be somewhat inferred by considering which of the benchmarks were attacked, other dimensions like attack implementation runtime or space complexity were not measurable (i.e., we did not offer a common computation platform). In some ways, we can interpret the results of this effort as a rough measure of what an attacker might be able to achieve end-to-end within $\sim$3 months, albeit with several caveats (e.g., teams were not working exclusively on the problem during the competition).

On Resources. The predilection towards combinational locking suggests that there is less familiarity with sequential locking in the community (at least, as represented by the participants). This raises a bigger question about what sort of resources and documentation could be provided to participants in future iterations to reduce any barriers to entry (and hopefully, encouraging more robust evaluation of the locking techniques). It would also be interesting to investigate what additional collateral material might support further evaluations of logic locking in “real-world” scenarios. For example, datasheets or other design documents as “plausible” supporting information that is accessible to an attacker; access to such information might enable a thorough examination of logic locking.

C. Perspectives on Application-level Directions

While the red/blue team exercise was underway, we supported an orthogonal exploration by *Team Z*. This team proposed an alternative attacker model that provides a promising new direction for logic locking research. Their approach aimed to quantify application-level ramifications of logic locking, considering each locked benchmark netlist as a small part of a larger and more complex system with a specific application. Because logic locking impacts the IC as a whole, application-level considerations provide important context for both combinational and sequential locking. With this context, architectural factors, such as error tolerance, and application-level factors, such as workload characteristics, must now be considered. This presents a holistic view of logic locking in practice.

To explore this attacker model, *Team Z* proposed a custom logic locking simulation framework, ObfusGEM. ObfusGEM integrates logic locked netlists within a cycle-accurate GEM5 model of a processor [71]. By running arbitrary workloads on this model and tracking the divergence between a locked and unlocked processor, the application-level impact of logic locking is quantified [72]. This provides a promising avenue to evaluate logic locking moving forward.

During this benchmarking, *Team Z* used ObfusGEM to integrate custom, small netlists, locked with SFLL-rem, into an x86 processor IC. Using this model, they showed that approximate keys, sufficient for the correct execution of a variety of software benchmarks, could be located, despite the presence of incorrectly keyed logic locking. While the locked netlists were of a smaller scale compared to the main benchmarking exercise, these results point at application-level context as a potential new direction for both attack and defense research, including hardware/software co-design and high-level synthesis, that can leverage application-level considerations to strengthen the security of logic locking. Thus, in future evaluations, the community should examine application-level considerations and the context they provide for combinational and sequential logic locking.

Several teams were able to recover the RLL keys for the combinational locking benchmarks, arguing that the remaining key inputs could be set randomly to retrieve an approximate circuit, with low error rate. This points to a well-known aspect of SFLL-based locking, where output corruption is limited to combat oracle-guided attacks. However, implications of an approximately recovered circuit is application dependent. Since different input/output pairs may have different levels of "importance" depending on the context, generalizing that the recovery of a circuit with a low error rate constitutes a successful attack may not be appropriate. How best to reflect application-level concerns in the next iteration is a direction to pursue.

D. Broader Reflections for Hardware Security

As mentioned in Section I, logic locking is one of several design-for-trust techniques. The lessons from our critical examination of the literature and benchmarking approach provide insights into the challenges faced in hardware security generally, and may in turn inform future efforts in evaluating alternative, emerging techniques in this area.

As we discussed in Section II, selecting appropriate benchmarks is crucial for characterizing techniques (e.g., with respect to overheads) and providing context for how effective an attack/defense is. In hardware security, more broadly, procuring benchmarks is not trivial. Unlike software, where there is ample (and production-quality) open-source resources, the hardware design community is not as populous (for example, there are $\geq$200K active Java projects on GitHub [73], whereas OpenCores contains $\sim$1K hardware IP-blocks [74]).

Having access to open-source tools might help with benchmark creation and evaluation. However, while open-source academic tools for hardware design exist (e.g., ABC [75] for logic synthesis), industrial electronic design automation (EDA) tools are less accessible, and this may preclude fully open-source logic locking tool-chains. Having said that, efforts such as OpenROAD [76], whose goal is an open-source RTL-to-GDSII flow, may change the future prospects.

Another area for future efforts include greater industry involvement, notwithstanding possibly complex legal IP issues or non-disclosure agreements, etc. For instance, the Hack@DAC [40] organizers were able to find a way for industry engineers to contribute meaningful hardware bugs to an open-sourced SoC design. While not security related, the long running ICCAD CAD contests [77] have shown success in bringing industry and academia together, such as the industry-provided benchmarks for physical design in [78]. How best to pursue this for design-for-trust techniques generally is an open question. Further, incorporating application-related context will be useful (e.g., identifying and quantifying important inputs and outputs). Industry could propose different benchmarks for characterizing different facets of the techniques, perhaps locking for datapaths, locking for sea-of-gates, or locking for SRAMs.

Our work has demonstrated the value of coordinated evaluation of hardware security techniques. With industry, government, and academic support, logic locking and other hardware security techniques can benefit from formal and ongoing evaluation. By making these processes regular and structured, researchers could submit new techniques on an ongoing basis for rigorous assessment. Such a process would increase confidence in hardware security technologies.

VIII. Future Outlook

Next Steps for Benchmarking. There are a number of areas that future iterations of community benchmarking can include for greater insights into the usability, practicality, and resilience of logic locking techniques. These include: (1) a common (possibly cloud-based) platform for comparing attack techniques: having a standard computing platform for launching attacks will allow better side-by-side comparisons of attack strategies with respect to their execution times, scalability, etc., (2) an even wider variety of benchmark circuits, possibly grouped by application domains (with this additional context provided to or withheld from the red teams depending on the threat model to be emulated), and (3) varying the amount of information provided to the attacker (e.g., application context). In our benchmarking exercise, we treated the combinational and sequential locking techniques in isolation—in the next iteration, it could be worth expanding the fielded defenses as well as investigate if we can gain more from combining techniques. The SFLL-rem technique was combined with RLL and the RLL-keys were broken. Perhaps other locking approaches can be mixed-in successfully.

Open-sourcing and Support Materials. For deeper scrutiny and wider adoption of logic locking, the natural next step will be fully open-source implementations of the locking techniques with detailed support materials (algorithm descriptions, user guides, etc.). This may pave the way for standardization, modeled after the NIST-style processes where the community-at-large publicly dissects the algorithm *and* its (reference) implementation. Increasing the visibility into the inner workings of locking techniques will increase confidence in correctness, and move the needle even closer towards Kerckhoff's principle in favor of security by obscurity. This is a challenge as locking techniques may use commercial tools as part of the tool flow. License costs may be prohibitive to some and non-disclosure agreements may hold back others.

Open Questions. While this paper explored shortcomings of the evaluation in the logic locking literature and attempted to address these through our foray in benchmarking, more open questions remain. These include ongoing questions about what constitutes a successful attack, i.e., how can we formalize meaningful notions of approximate recovery or application-level concerns? More work could also look to separating analysis of a locking technique from attacks that arise from a flawed implementation of the technique. Furthermore, is it possible to definitively conclude that a defensive technique has "graduated" from this process? This will inform the steps that could be taken after a technique succeeds at this process, in terms of potential impacts on wider adoption or public policy.

Concluding Remarks. In this paper, we prepared, ran, and reflected on the first benchmarking effort in logic locking. Through this process we worked towards leveling the playing field where defenders and attackers were given the opportunity to "put their best foot forward". As it stands, SFLL-rem and State-based locking remain "unbroken" based on this benchmarking effort, with the fielded attacks offering only partial success. Our efforts produced a timely snapshot of the current state-of-the-art in logic locking for digital design IP protection. As these techniques mature and new attacks and defenses emerge, the lessons learned from this community effort will provide a foundation for future endeavors.

Acknowledgments

B. Tan is supported in part by ONR Award # N00014-18-1-2058. R. Karri is supported in part by National Science Foundation (NSF) Grant #1513130, # N00014-18-1-2058 and NYU/NYUAD Center for Cyber Security. O. Sinanoglu, N. Limaye and A. Sengupta are supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant HR001116S0001-FP39, and in part by the NYUAD Center for Cyber Security. S. Bhunia and M. Rahman were supported in part by the DARPA OMG Program grant. Z. Han, A. Benedetti, L. Brignone, M. Yasin, and J. Rajendran are supported by the Office of Naval Research (ONR Award #N00014-18-1-2058), the National Science Foundation (NSF CNS-1822848), and the Defense Advanced Research Projects Agency (DARPA HR001116S0001-FP39). This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA)

Disclaimer. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] J. L. Hennessy and D. A. Patterson, "A New Golden Age for Computer Architecture," *Commun. ACM*, vol. 62, no. 2, p. 4860, Jan. 2019.

[2] M. Yasin, J. J. Rajendran, and O. Sinanoglu, *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Cham, Switzerland: Springer International Publishing, 2020.

[3] A. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *IEEE/ACM Design Automation Conference*, 1998, pp. 776–781.

[4] Rambus, "Circuit Camouflage Technology," https://www.rambus.com/security/cryptofirewall-cores/circuit-camouflage-technology, 2020, [May 21, 2020].

[5] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security Analysis of Integrated Circuit Camouflaging," in *ACM/SIGSAC Conference on Computer & Communications Security*, 2013, pp. 709–720.

[6] J. Baukus, L. Chow, R. Cocchi, and B. Wang, "Method and Apparatus for Camouflaging a Standard Cell based Integrated Circuit with Micro Circuits and Post Processing," 2012, uS Patent no. 20120139582.

[7] R. Jarvis and M. McIntyre, "Split Manufacturing Method for Advanced Semiconductor Circuits," 2007, US Patent 7,195,931.

[8] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, "Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation," in *USENIX Conference on Security*, 2013, pp. 495–510.

[9] Y. Alkabani and F. Koushanfar, "Active Hardware Metering for Intellectual Property Protection and Security," in *USENIX Security*, 2007, pp. 291–306.

[10] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 51–63, 2012.

[11] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *Proc. Conf. Design, Automation and Test in Europe (DATE)*. New York, NY, USA: ACM, 2008, pp. 1069–1074.

[12] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proc. Design, Automation Test in Europe Conf. (DATE)*, Mar. 2012, pp. 953–958, iSSN: 1530-1591.

[13] ——, "Security analysis of logic obfuscation," in *Proc. 49th Annu. Design Automation Conf (DAC)*, Jun. 2012, pp. 83–89.

[14] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2016, pp. 236–241.

[15] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," Cryptology ePrint Archive, Report 2017/761, 2017, http://eprint.iacr.org/2017/761.

[16] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice," in *Proc. ACM SIGSAC Conf. Computer Communications Security (CCS)*. Association for Computing Machinery, Oct. 2017, pp. 1601–1618.

[17] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly stripping functionality for logic locking: A fault-based perspective," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–1, 2020.

[18] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug 2014.

[19] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain," *IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[20] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware Trojans: Lessons Learned after One Decade of Research," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, pp. 6:1–6:23, May 2016.

[21] J. Roy, F. Koushanfar, and I. L. Markov, "Ending Piracy of Integrated Circuits," *Computer*, vol. 43, no. 10, pp. 30–38, 2010.

[22] R. S. Chakraborty and S. Bhunia, "HARPOON: an obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, 2009.

[23] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, "What to Lock? Functional and Parametric Locking," in *Proc. Great Lakes Symp. VLSI (GLVSI)*, May 2017, pp. 351–356.

[24] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 85–90.

[25] M. Yasin, J. J. Rajendran, and O. Sinanoglu, *A Brief History of Logic Locking*. Cham, Switzerland: Springer International Publishing, 2020, pp. 17–31.

[26] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. Washington, DC, USA: IEEE, May 2015, pp. 137–143. [Online]. Available: http://ieeexplore.ieee.org/document/7140252/

[27] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *HOST*. IEEE, 2017, pp. 95–100.

[28] Y. Shen, A. Rezaei, and H. Zhou, "SAT-based bit-flipping attack on logic encryptions," in *2018 Design, Automation Test in Europe Conf. (DATE)*, Mar. 2018, pp. 629–632, iSSN: 1558-1101.

[29] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *GLSVLSI*. ACM, 2017, pp. 179–184.

[30] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-based attack on cyclic logic encryptions," in *2017 IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 2017, pp. 49–56, iSSN: 1558-2434.

[31] L. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," in *2019 Design, Automation Test in Europe Conf. (DATE)*, Mar. 2019, pp. 540–545, iSSN: 1530-1591.

[32] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and Hardware Trojans," in *2014 IEEE 20th Int. On-Line Testing Symp. (IOLTS)*, Jul. 2014, pp. 49–54, iSSN: 1942-9401.

[33] S. Engels, M. Hoffmann, and C. Paar, "The End of Logic Locking? A Critical View on the Security of Logic Locking," Cryptology ePrint Archive, Report 2019/796, 2019, http://eprint.iacr.org/2019/796.

[34] M. T. Rahman, S. Tajik, M. S. Rahman, M. Tehranipoor, and N. Asadizanjani, "The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes," Cryptology ePrint Archive, Report 2019/719, 2019, https://eprint.iacr.org/2019/719, *to appear at HOST 2020*.

[35] NIST, "CSRC Presentation: PQC: Announcement and outline of NIST's Call for Submissions | CSRC." [Online]. Available: https://csrc.nist.gov/Presentations/2016/Announcement-and-outline-of-NIST-s-Call-for-Submis

[36] "A Comprehensive List of Cybersecurity Competitions." [Online]. Available: https://www.cybersecuritydegrees.com/faq/comprehensive-list-of-cyber-security-competitions/

[37] A. Fasano, T. Leek, B. Dolan-Gavitt, and J. Bundt, "The Rode0day to Less-Buggy Programs," *IEEE Security Privacy*, vol. 17, no. 6, pp. 84–88, Nov. 2019.

[38] "DARPA Challenge Tests AI as Cybersecurity Defenders." [Online]. Available: https://spectrum.ieee.org/tech-talk/computing/software/darpa-challenge-tests-ai-as-cybersecurity-defenders

[39] "CSAW Embedded Systems Challenge." [Online]. Available: https://web.archive.org/web/20121114001755/http://www.poly.edu/csaw2012/csaw-embedded

[40] G. Dessouky, D. Gens, P. Haney, G. Persyn, A. Kanuparthi, H. Khattri, J. M. Fung, A.-R. Sadeghi, and J. Rajendran, "HardFails: Insights into Software-Exploitable Hardware Bugs," in *USENIX Security Symposium*, 2019, pp. 213–230. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/dessouky

[41] C. Clavier, J.-L. Danger, G. Duc, M. A. Elaabid, B. Grard, S. Guilley, A. Heuser, M. Kasper, Y. Li, V. Lomn, D. Nakatsu, K. Ohta, K. Sakiyama, L. Sauvage, W. Schindler, M. Stttinger, N. Veyrat-Charvillon, M. Walle, and A. Wurcker, "Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest," *Journal of Cryptographic Engineering*, vol. 4, no. 4, pp. 259–274, Nov. 2014.

[42] "Trust-Hub." [Online]. Available: https://www.trust-hub.org/home

[43] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic Obfuscation for Creating SAT-Unresolvable Circuits," in *Proc. Great Lakes Symp. VLSI (GLSVLSI)*. Banff, Alberta, Canada: ACM Press, 2017, pp. 173–178.

[44] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu, "ATPG-based cost-effective, secure logic locking," in *Proc. IEEE 36th VLSI Test Symp. (VTS)*, Apr. 2018, pp. 1–6, iSSN: 2375-1053.

[45] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," in *Proc. Design, Automation Test in Europe Conf. DATE)*. Florence, Italy: IEEE, Mar. 2019, pp. 936–939.

[46] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of Anti-SAT," in *22nd Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Jan. 2017, pp. 342–347.

[47] ——, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2017.

[48] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against All Known Logic Locking Attacks," in *Cryptographic Hardware and Embedded Systems CHES 2017*, ser. Lecture Notes in Computer Science, W. Fischer and N. Homma, Eds. Cham: Springer International Publishing, 2017, pp. 189–210.

[49] Y. Zhang, P. Cui, Z. Zhou, and U. Guin, "Tga: An oracle-less and topology-guided attack on logic locking," in *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, 2019, pp. 75–83.

[50] A. Chakraborty, N. G. Jayasankaran, Y. Liu, J. Rajendran, O. Sinanoglu, A. Srivastava, Y. Xie, M. Yasin, and M. Zuzak, "Keynote: A Disquisition on Logic Locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–1, 2019.

[51] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," *IEEE Des. Test. Comput.*, vol. 16, no. 3, pp. 72–80, Jul. 1999. [Online]. Available: https://doi.org/10.1109/54.785838

[52] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," 1991, Technical Report MCNC P.O. Box 12889, Research Triangle Park, NC 27709.

[53] "OpenSPARC T1 Processor." [Online]. Available: http://www.oracle.com/technetwork/systems/opensparc/opensparc-t1-page-1444609.html

[54] L. Basto, "First results of ITC'99 benchmark circuits," *IEEE Des. Test. Comput.*, vol. 17, no. 3, pp. 54–59, Sep. 2000.

[55] V. V. Menon, G. Kolhe, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo, "System-Level Framework for Logic Obfuscation with Quantified Metrics for Evaluation," in *2019 IEEE Cybersecurity Development (SecDev)*, Sep. 2019, pp. 89–100.

[56] "Common Evaluation Platform — MIT Lincoln Laboratory." [Online]. Available: https://www.ll.mit.edu/r-d/projects/common-evaluation-platform

[57] Y. Hu, V. V. Menon, A. Schmidt, J. Monson, M. French, and P. Nuzzo, "Security-driven metrics and models for efficient evaluation of logic encryption schemes," in *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*. La Jolla, California: Association for Computing Machinery, Oct. 2019, pp. 1–5.

[58] M. T. Rahman, M. S. Rahman, H. Wang, S. Tajik, W. Khalil, F. Farahmandi, D. Forte, N. Asadizanjani, and M. Tehranipoor, "Defense-in-depth: A recipe for logic locking to prevail," *Integration*, vol. 72, pp. 39–57, May 2020.

[59] R. Schlangen, R. Leihkauf, U. Kerst, C. Boit, R. Jain, T. Malik, K. Wilsher, T. Lundquist, and B. Kruger, "Backside E-Beam Probing on Nano scale devices," in *2007 IEEE Int. Test Conf. (ITC)*, Oct. 2007, pp. 1–9, iSSN: 2378-2250.

[60] C. Boit, C. Helfmeier, D. Nedospasov, and A. Fox, "Ultra high precision circuit diagnosis through seebeck generation and charge monitoring," in *Proc. 20th IEEE Int. Symp. Physical and Failure Analysis of Integrated Circuits (IPFA)*, Jul. 2013, pp. 17–21, iSSN: 1946-1550.

[61] U. Kindereit, "Fundamentals and future applications of Laser Voltage Probing," in *2014 IEEE Int. Reliability Physics Symp.*, Jun. 2014, pp. 3F.1.1–3F.1.11, iSSN: 1938-1891.

[62] P. Song, F. Stellari, D. Pfeiffer, J. Culp, A. Weger, A. Bonnoit, B. Wisnieff, and M. Taubenblatt, "MARVEL Malicious alteration recognition and verification by emission of light," in *2011 IEEE International Symp. Hardware-Oriented Security and Trust (HOST)*, Jun. 2011, pp. 117–121.

[63] P. Song, F. Stellari, B. Huott, O. Wagner, U. Srinivasan, Y. Chan, R. Rizzolo, H. Nam, J. Eckhardt, T. McNamara, C.-L. Tong, A. Weger, and M. McManus, "An advanced optical diagnostic technique of IBM z990 eServer microprocessor," in *IEEE Int. Cong Test (ITC)*, Nov. 2005, pp. 9 pp.–1235, iSSN: 2378-2250.

[64] F. Stellari, P. Song, M. Villalobos, and J. Sylvestri, "Revealing SRAM memory content using spontaneous photon emission," in *2016 IEEE 34th VLSI Test Symp. (VTS)*, Apr. 2016, pp. 1–6, iSSN: 2375-1053.

[65] S. M. Mansfield and G. S. Kino, "Solid immersion microscope," *Applied Physics Letters*, vol. 57, no. 24, pp. 2615–2616, Dec. 1990, publisher: American Institute of Physics. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.103828

[66] U. Kindereit, A. J. Weger, F. Stellari, P. Song, H. Deslandes, T. Lundquist, and P. Sabbineni, "Near-infrared photon emission spectroscopy of a 45 nm soi ring oscillator," in *2012 IEEE International Reliability Physics Symposium (IRPS)*, 2012, pp. 2D.2.1–2D.2.7.

[67] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse engineering digital circuits using functional analysis," in *Proc. Design, Automation Test in Europe Conf. (DATE)*, Mar. 2013, pp. 1277–1280.

[68] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Proc. 49th Annu. Design Automation Conf. (DAC)*, Jun. 2011, pp. 333–338.

[69] D. Duvalsaint, Z. Liu, A. Ravikumar, and R. D. Blanton, "Characterization of Locked Sequential Circuits via ATPG," in *2019 IEEE Int. Test Conf. Asia (ITC-Asia)*, Sep. 2019, pp. 97–102.

[70] "Atalanta: an efficient forward fault simulation ATPG for combinational circuits," 1991, Technical Report, 93-12, Department of Electrical Engineering, Virginia Polytechnic Institute and State University.

[71] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[72] M. Zaman, A. Sengupta, D. Liu, O. Sinanoglu, Y. Makris, and J. J. V. Rajendran, "Towards provably-secure performance locking," in *Proc. Conf. Design, Automation and Test in Europe (DATE)*, Mar. 2018, pp. 1592–1597, iSSN: 1558-1101.

[73] "GitHut - Programming Languages and GitHub." [Online]. Available: https://githut.info/

[74] "Statistics :: Opencores." [Online]. Available: https://opencores.org/about/statistics

[75] "Abc: System for sequential logic synthesis and formal verification." [Online]. Available: https://github.com/berkeley-abc/abc

[76] T. Ajayi, V. A. Chhabria, M. Fogaa, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu, "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project," in *Proc. 56th Annu. Design Automation Conf. (DAC)*. ACM, Jun. 2019, pp. 1–4.

[77] "2020 cad contest @ iccad." [Online]. Available: http://iccad-contest.org/2020/

[78] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ISPD2005 placement contest and benchmark suite," in *Proc. Int. Symp. Physical Design (ISPD)*. ACM Press, 2005, p. 216.

APPENDIX

While some red teams proposed new attacks, others used attacks published in the literature. None of the attacks successfully recovered the entire key. This section provides further details on the attack attempts (in no specific order).

ATPG-based (Oracle-guided) *Team A* applied the sensitization attack [13] on the combinational locking benchmarks. In this attack, automatic test pattern generator (ATPG)[4] is used to detect a fault at a key input, with the other keys tied to a don't-care. If a test pattern is found, this indicates that there is a path from the key input to a primary output that does not require setting the other keys. The test pattern is applied to the oracle to propagate the value of the key input to the primary output. To recover the SFLL-rem key bits, a different approach based on fault equivalence and dominance was attempted. This attack looks for a fault that is equivalent or dominant to the fault that was removed. While the team identified the primary inputs connected to the key inputs, they failed to recover the SFLL-rem key. *They recover 100% of RLL key bits and none of the SFLL-rem key bits.*

The team used an ATPG-based method to attack the sequential locking benchmarks, based upon [69]. To attack a locked sequential circuit, the ATPG-based attack unrolled the circuit (as scan-chain access was not provided). Unrolling a sequential circuit increases the attack complexity exponentially and this complexity increases with the number of registers. In order to deploy a successful attack, the circuit needs to be fully unrolled. The team incorrectly guessed the number of cycles to unroll to be 5 cycles. This led to an incomplete unrolling and a failure to recover the complete key.

[4]ATPG is used to test circuits to detect manufacture-time faults such as a wire stuck-at-zero and stuck-at-one.

Hamming Distance (HD)-based Attack (Oracle-guided) *Team B* proposed a divide-and-conquer approach to attack the combinational locking. There are three steps to perform this attack: identify the type of key inputs, strip partial RLL key inputs, and launch the attack.

1) Split the locked netlist into individual logic cones (ILCs).
2) Count the number of key bits in each cone to identify the type of protection for each ILC and the type of each key input. Since they find all ILCs and each cone's protection type, they can find the RLL key value.
3) Select all RLL cones and merge them into a netlist with fewer primary outputs (compared to the full netlist). The SAT attack on this netlist returns a valid RLL key with partial RLL key inputs. Applying this key to the original locked netlist produces a simplified locked circuit.

To attack the SFLL-rem lock, they pick one cone which is only protected by SFLL, as shown in Algorithm 1. First, they extract the functionality stripped circuit (FSC) from the locked cone. From this they collect a set of candidate protected input patterns (PIPs) whose Hamming distances are no greater than a threshold d to at least one PI in the FSC's reduced PI table. The PIP candidates are fed into the oracle and the FSC to identify differences in the output, thus verifying that the input pattern is indeed protected. If they can find one verified PIP, they use this PIP as the first input pattern into the SAT-based attack and find the correct key from the attack. This key is valid for other locked cones. *This team recovered most of the RLL key bits and none of the SFLL-rem key bits.*

Algorithm 1 Hamming Distance (HD)-based Attack

```
Require: A SFLL-fault cone $\mathcal{C}_{locked}$, oracle $\mathcal{O}$, parameter $d$
Ensure: Correct key $key_c$
1: $\mathcal{C}_{FSC} \leftarrow$ extract_FSC($\mathcal{C}_{locked}$)
2: $Reduced\_PIT \leftarrow$ extract_PI_table($\mathcal{C}_{FSC}$)
3: $PIP_{cand} \leftarrow \{p | HD(p, pi) \leq d, \exists pi \in Reduced\_PIT\}$
4: for $p \in PIP_{cand}$ do
5:     if $\mathcal{O}(p) \neq \mathcal{C}_{FSC}(p)$ then
6:         $key_c \leftarrow$ SAT_simulation($\mathcal{C}_{locked}, \mathcal{O}, p$)
7:         return $key_c$
8:     end if
9: end for
```

Automated SAT Attack (Oracle-guided) *Team C* attempted a divide-and-conquer approach by dividing the circuit into smaller logic cones, starting with primary outputs and identifying the fan-in logic. The team automated this structural-level analysis and performed a SAT attack on the logic cones, focusing on those cones with fewer key inputs. The circuit is divided into individual logic cones. An analysis of the cones reveals that some outputs depend on only one key input and a relatively small number of inputs. As an intermediate step, they attempted to find DIPs on a small logic cone with one key input, 33 primary inputs, and one primary output. The

Algorithm 2 Bit-flipping Attack

```
Require: Encrypted circuit C(X, K, Y) and oracle $eval$.
Ensure: Encryption key $K_c$.
 1: i = 1
 2: $F_1 = C(X, K_1, Y_1) \wedge C(X, K_2, Y_2)$
 3: Fixing SFLL-rem keys in $K_1$ and $K_2$ to a random value
 4: while $sat[F_i \wedge (Y_1 \neq Y_2)]$ do
 5:     $X_i = sat\_assignment_X(F_i \wedge (Y_1 \neq Y_2))$
 6:     $Y_i = eval(X_i)$
 7:     $F_{i+1} = F_i \wedge C(X_i, K_1, Y_i) \wedge C(X_i, K_2, Y_i)$
 8:     $i = i + 1$
 9: end while
10: $K_c = sat\_assignment_{K1}(F_i)$
```

SAT solver was unable to solve the reduced circuit, so a key sensitization attack on the smaller logic cone was performed. This attack chooses an arbitrary set of inputs, executes the oracle, collects the targeted output, and runs the SAT solver with the key input as the unknown. The SAT solver produced an output for the single unknown value. After finding some keys from logic cones with a single key input, the team targeted logic cones with increasing numbers of key inputs and one unknown key input. *The team recovered <50% of RLL key bits and no SFLL-rem key bits.*

Sub-circuit SAT Attack (Oracle-guided) *Team D* launched a SAT attack on combinational locking. They used two strategies to find a sub-circuit on which to apply SAT attack.

1) They find an output which depends on one key input. Then, the SAT attack is applied for the sub-circuit. If the SAT solver iterates twice to find the value of a key, it indicates that this value is correct.
2) They uncover a key input which propagates to only one output. A SAT attack is mounted on the sub-circuit involving the cone of influence of this output. The resulting key is correct as it depends only on that output and this value can be used to find additional key inputs recursively.

They recovered <50% of the RLL key bits and none of the SFLL-rem key bits.

Redundancy attack (Oracle-less) *Team E* used the redundancy attack [31] on the combinational locking benchmarks. The attack is based on the observation that key gates modifies the netlist after synthesis and therefore produce arbitrary invalid design options that fail to adhere to certain design principles such as redundancy removal. The attack dismantles the complexity of the key space by deciphering key bits individually or in pairs. Since RLL key bits are inserted in the middle of the netlist, they have stronger local impacts on the redundancy level of nearby regions where a single modification to the intertwined re-convergent structures alone results in untestable faults. On the other hand, incorrect key values for SFLL result in untestable faults at the convergence point of the functionality stripped circuit and recovery unit. The removal of such redundant faults would make certain output bits completely unprotected by SFLL, thus proving the invalidity of the key assignment. *The team recovered >50% of RLL key bits and <50% of the SFLL-rem key bits.*

Bit-Flipping Attack (Oracle-guided) *Team F* adopted the bit-flipping attack [28] on the combinational locking benchmarks based on. The key bits of RLL and SFLL-rem are separated by fixing the key values to random values with the Hamming distance equal to one, and counting the number of Distinguishing Input Patterns (DIPs). DIPs are used to differentiate between the design outputs when different key values are applied. Since the error rate of SFLL-rem is exponentially low, the protected input patterns are rarely applied even if the SFLL-rem key is wrong. Thus, they randomly fix the SFLL-rem key and solve the RLL key by applying the SAT attack [26]. This attack recovers all the RLL key bits in under a minute for the largest benchmark. Algorithm 2 shows the methodology. *They recovered all RLL key bits and none of the SFLL-rem key bits.*

Unit Function Search Attack (Oracle-less attack) *Team G* mounted the Unit Function Search attack on the combinational locking benchmarks [49]. If one or more key gates are placed in an instance of repeated unit function (*UF*) during the locking of a circuit, the original netlist can be recovered by searching the equivalent unit functions (*EUFs*) with all hypothesis keys. The hypothesis key bit will be the actual secret key bit if a match is found. The attack fails when the search fails to find a match with all hypothesis keys.

This attack uses an efficient depth-first search to find the EUFs in a locked netlist. Searching the EUFs in the netlist is equivalent to the subgraph isomorphism problem. Hence, they convert the netlist to a directed graph, where each gate in the netlist is a vertex, and each wire is an edge. For each EUF, the search algorithm traverses the generated graph to check for the existence of the same structure.

Since each key bit is targeted individually, the average time to determine a secret key bit is in the order of seconds. The initial version of this attack in [49] targeted RLL presented in [11], where there is no inter-dependency among key bits. *This attack recovered <50% of RLL key bits, and no SFLL-rem key bits.*