Risk-Aware Cost-Effective Design Methodology for Integrated Circuit Locking

Yinghua Hu, Kaixin Yang, Subhajit Dutta Chowdhury, Pierluigi Nuzzo

Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA
{yinghuah, kaixinya, duttacho, nuzzo}@usc.edu

***Abstract*—We introduce a systematic framework for logic locking of integrated circuits based on the analysis of the sources of information leakage from both the circuit and the locking scheme and their formalization into a notion of risk that can guide the design against existing and possible future attacks. We further propose a two-level optimization-based methodology to generate locking strategies minimizing a cost function and balancing security, risk, and implementation overhead, out of a collection of locking primitives. Optimization results on a set of case studies show the potential of layering multiple locking primitives to provide high security at significantly lower risk.**

***Index Terms*—Hardware security, logic locking, risk modeling**

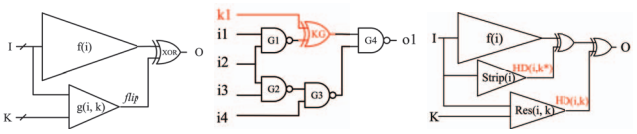
I. Introduction

Logic Locking (LL) has gained significant attention as a promising solution against reverse engineering of integrated circuits (ICs) by mistrusted, third-party players. The goal of LL is to modify the logic of a circuit by infusing *programmability* in the design, e.g., by adding extra components and a set of key inputs, such that the circuit functionality can be retrieved only if the correct configuration key is applied. The mechanisms to achieve programmability have become increasingly more sophisticated and resilient over the years, from XOR-based [1], [2] to point-function-based insertion methods [3]–[7]. However, these advances have mostly been driven by the conception of increasingly stronger attacks, and the need to circumvent them, possibly leaving on the table other sources of vulnerability that could be still exploited by newer attacks.

Recently, a set of rigorous *threat models* and *security-driven metrics* [6], [8]–[10] has been introduced as a foundation to guide the design of resilient LL methods, by rigorously capturing design objectives such as the "level of protection" and the "attack resilience." However, in the absence of certainty about the level of security of a given construction, these models fall short of characterizing residual sources of vulnerability and "information leakage" associated with the circuit and the locking scheme upon deployment. On the other hand, the functional and structural "signatures" of a circuit can be better obfuscated, and attack hardness increased, by replacing portions of the original circuit with look-up tables (LUTs) or similar structures [8], [10], [11]. These structures tend, however, to be expensive, calling for new design methods that can appropriately balance the implementation cost with the desired level of protection, attack resilience, and risk.

This paper takes a first step toward addressing this challenge. By drawing from concepts in Boolean analysis and learning theory, we define a set of *risk metrics* that quantify the likelihood that a circuit or LL scheme disclose information about their function or structure, exposing intrinsic circuit vulnerabilities which can be exploited by an existing or hypothetical attack.

This work was partially supported by AFRL and DARPA under agreement number FA8650-18-1-7817.



(a) (b) (c)

Fig. 1. Schematic of (a) a generic LL scheme, (b) XBI, and (c) SFLL-HD.

Based on these metrics, we present a *two-level optimization-based methodology* combining simulated annealing (SA) [12] and mixed integer linear programming (MILP) [13] to find an LL construction, out of a library of LL primitives, that satisfies a set of security and risk driven requirements while minimizing a cost function. To the best of our knowledge, this is the first automated optimization-based framework for the design of LL schemes that systematically captures security-driven properties, implementation overhead, and residual risk. Optimization results on a set of ISCAS benchmark circuits as well as a case study on the Common Evaluation Platform (CEP) [14] show the effectiveness of the proposed approach.

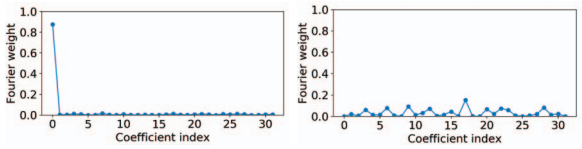
II. Background

We first provide background concepts on the threat models and security metrics adopted in this paper.

Circuit Representation. Given a combinational circuit with primary input port set I and primary output port set O, the circuit function $f : \mathbb{B}^{|I|} \rightarrow \mathbb{B}^{|O|}$ maps input vectors to the corresponding output vectors. A logic locking scheme changes f to a new function $f' : \mathbb{B}^{|I|} \times \mathbb{B}^{|K|} \rightarrow \mathbb{B}^{|O|}$ by adding extra key input ports K to the circuit. A key vector k^* is the correct key if $\forall\ i \in \mathbb{B}^{|I|}$, $f(i) = f'(i, k^*)$ holds. A generic functional model for logic locking decomposes $f'(i, k)$ as $f'(i, k) = f(i) \oplus g(i, k)$, where $g(i, k)$ represents the effect of the logic locking scheme [9]. A schematic of a locked circuit is shown in Fig. 1(a), where the $flip$ signal is the output of $g(i, k)$. If $g(i, k) = 1$, the circuit output value for input i and key k is incorrect, i.e., $f'(i, k) \neq f(i)$. Otherwise, the circuit output is correct, i.e., $f'(i, k) = f(i)$. A circuit netlist can be modeled as a labelled directed graph $G = (V, E)$, where V is the set of graph vertices, representing input-output (I/O) ports, logic gates, or sub-modules, and E is the set of edges, representing interconnections. The edges have directions indicating the order of signal propagation during logic simulation.

Attack Assumptions. In line with the literature on logic locking, we assume the following two resources are available to attackers: (i) the netlist of the locked circuit, and (ii) a black-box circuit, i.e., an *oracle*, providing the I/O pairs for the correct circuit function.

Security Metrics. We quantify the security of our designs using two previously proposed metrics [9], namely, *functional corruptibility* (E_{FC}) and *SAT-attack resilience* (t_{SAT}).



(a) (b)

Fig. 2. Fourier spectrum of (a) 5-input AND function and (b) 5-input pseudo-random function.

III. A Notion of Risk for Logic Locking

An LL scheme that is resilient to attacks and provides sufficient protection can still exhibit sources of vulnerability stemming from the circuit function or structure. We characterizes these sources via the following risk models.

Boolean Learnability Risk. A worst-case brute-force attack needs to perform $2^{|I|}$ queries to the oracle to reconstruct the correct circuit function. However, some Boolean functions can be well approximated with much fewer queries. We quantify the risk associated with the learnability of a Boolean function by resorting to the "concentration" of its Fourier spectrum from Boolean analysis.

Any Boolean function $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$, redefined, with a slight abuse of notation, on the set $\{-1, 1\}^n$ and with values in $\{-1, 1\}$, where -1 and 1 denote `false` and `true`, respectively, can be uniquely represented by its Fourier expansion as a real, multi-linear polynomial [15] $f(x) = \sum_{S\subseteq[n]} \hat{f}(S) \prod_{i\in S} x_i$, where $[n] = \{1, 2, ..., n\}$. The collection of all Fourier coefficients is the Fourier spectrum of f. The square of a Fourier coefficient is a Fourier weight. The sum of all Fourier weights of f is always 1. Informally, if the Fourier spectrum is concentrated over a small collection of subsets $S \subseteq [n]$, then learning those coefficients is enough to get a close approximation of f. This notion is formalized by the following results [15].

***Definition** 1 (Spectrum Concentration):* Suppose $\mathcal{S}$ is a collection of subsets $S \subseteq [n]$ and ϵ a non-negative real number. Then, the Fourier spectrum is ϵ-concentrated on $\mathcal{S}$ if $\sum_{S\subseteq[n],S\notin\mathcal{S}} \hat{f}(S)^2 \leq \epsilon$.

***Theorem** 1:* Assume a learning algorithm $\mathcal{A}$ has (at least) random query access to the target $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$. If $\mathcal{A}$ can identify a collection $\mathcal{S}$ of subsets such that the Fourier spectrum of f is $\epsilon/2$-concentrated, then, by using $poly(|\mathcal{S}|, n, 1/\epsilon)$ additional time, $\mathcal{A}$ can with high probability provide a hypothesis h that is ϵ-close to f.

We are now ready to define the Boolean learnability risk.

***Definition** 2 (Boolean Learnability Risk):* Given a logic cone (single-output combinational circuit) function f and a support parameter s, let $\mathcal{S}$ be a collection of Fourier coefficients, we define the Boolean learnability risk as

$$R_{b,s} = \max_{|\mathcal{S}|=s} \left\{ \sum_{S\in\mathcal{S}} \hat{f}(S)^2 \right\}.$$

If there exists some $\mathcal{S}$ with (small) cardinality s, on which the spectrum of f is ϵ-concentrated, then $R_{b,s}$ is at least $1-\epsilon$. $R_{b,s}$ can be computed by using the Goldreich-Levin algorithm [15] to find the most significant Fourier coefficients of f. Fig. 2 compares the Fourier spectrum of a 5-input AND function with a 5-input pseudo-random function. Suppose $s = 1$; then, $R_{b,1}$ for the AND function and the pseudo-random function are 0.86 and 0.15, respectively. Intuitively, The AND function has a highly concentrated spectrum, which makes it easier to approximate than the pseudo-random function.

Isolation Risk. LL creates functional corruptibility via the addition of key-controlled logic blocks capable of influencing the values of some of the nodes in the original circuit. The *isolation risk* R_i quantifies the difficulty of retrieving the correct circuit function by isolating those key-controlled blocks from the locked circuit.

***Definition** 3 (Isolation Risk):* Given a locked circuit, suppose c is the number of nets connecting the key-controlled blocks to the original circuit block. We define the isolation risk as $R_i = (1 - E_{FC,r})2^{-c}$, where $E_{FC,r}$ is the residual functional corruptibility (error rate) after isolating the key-controlled logic and 2^{-c} is the probability of finding the correct assignment of the c control signals, under the assumption that they are uniformly distributed.

For example, in SARLock [3], whose schematic is similar to the one in Fig. 1(a), there is only one wire connecting the output of the key-controlled block to the original circuit, so $R_i = 0.5$, suggesting high risk for removal attacks. Conversely, SFLL [6] pre-corrupts some of the outputs of $f(i)$ in Fig. 1(a) and relies on $g(i, k)$ to restore them, so $E_{FC,r}$ is larger than zero, which makes R_i of SFLL lower than the one of SARLock.

Repetition Risk. The implementation of an LL scheme and, specifically, the key-controlled block can reveal critical information about the correct key. The repetition risk aims to quantify the likelihood that different key ports with similar structure can be correlated to narrow down the range of possible key values.

***Definition** 4 (Repetition Risk):* Given a locked circuit $f'(i, k)$, suppose that an algorithm $\mathcal{A}$ can efficiently group the key input ports K into m clusters, i.e., $G_1, G_2, \ldots, G_m$, based on structural similarities in the fan-out cones of different key input ports. Let w_i, $i \in \{1, \ldots, m\}$, be the weight of cluster G_i. If the correct key bits for all key ports in G_i are the same, which means that similar structures correlate to the same correct key values, then w_i is one. Otherwise, we assign w_i to be $|G_i|$. We then define the repetition risk R_r as $R_r = 2^{-\sum_{i=1}^{m} w_i}$.

For example, XOR or XNOR gates can be inserted as key gates in the XOR-based insertion (XBI) method [1], [2], as shown in Fig. 1(b). However, all the key ports can be grouped into two clusters based on whether the key gates are XOR or XNOR, leading to a correct key bit of zero or one, respectively. In this case, $w_1 = w_2 = 1$, hence $R_r = 0.25$. While performing logic re-synthesis can decrease the structural correlation observed in Fig. 1(b) by using different implementations for the XOR/XNOR gates, the transformation performed by commercial synthesis tools may be partially reverted [16], or it may be possible to identify and correlate the logic function of the fan-out cone of each key port [17]. On the other hand, in SFLL-HD [6], shown in Fig. 1(c), two Hamming Distance (HD) comparators are implemented as part of the LL scheme. Since the only block in the fan-out cone of the key ports is $Res(i, k)$ and there is no *a priori* correlation between the structure of each key port and the correct key value, we can only report one cluster containing all the key ports, with a weight of $|K|$. In this case, the repetition risk is lower than that of XBI.

Identification Risk. The identification of the circuit structure can disclose important details about the circuit, which allow limiting the number of Boolean functions that can be represented by the specific structure. However, if multiple circuit structures can be obtained for different key values, it becomes

harder for the attacker to efficiently limit the space of possible functions that can be represented by the original circuit. We capture this aspect via the identification risk.

***Definition** 5 (Identification Risk):* For a given locked circuit netlist the identification risk R_{id} is inversely proportional to the number of circuit structures $|\mathcal{C}|$ achievable by key-configurable routing of the locked netlist, i.e., $R_{id} = \frac{1}{|\mathcal{C}|}$.

For example, the XOR gate `KG` in Fig. 1(b) decides whether to flip the value of `G1` but does not affect the routing configuration of the circuit. Hence, the identification risk for XBI is 1. On the other hand, given a universal circuit with d layers, each layer containing m 2-input LUTs, it is possible to achieve w^{2wd} different structures, a much higher number than that in XBI, by also programming the routing configuration.

IV. Risk-Aware Optimization-Based Locking

We detail an optimization-based methodology that can balance security objectives and residual risk with overhead.

Problem Formulation. We assume that a library $\mathcal{L}$ of LL primitives is available, where each primitive l is described by a set of configuration parameters p_l, and a set of models providing the security metrics q_l, risk metrics r_l, and overhead metrics o_l as a function of p_l via the following mappings:

$$q_l = m_{q,l}(p_l),\ r_l = m_{r,l}(p_l),\ o_l = m_{o,l}(p_l). \tag{1}$$

For example, $p_{SARLock}$ includes the key size and $m_{q,SARLock}$ computes E_{FC} and t_{SAT} as a function of the key size.

Since multiple instances of the same primitive can be used in the design, we define a binary variable matrix $s \in \mathbb{B}^{|\mathcal{L}|\times n}$ such that n is the maximum allowed number of instances of any primitive in the design, and $s_{ij} = 1$ if and only if the jth instance of primitive i is used. Correspondingly, we use p_{ij}, q_{ij}, r_{ij}, o_{ij} to define the configuration parameters, security, risk, and overhead metrics for instance j of primitive i, respectively.

Primitives are composed according to pre-defined rules. For example, the cumulative effect of multiple point-function-based primitives can be obtained by computing the logical disjunction of all output signals of the different primitives and connecting it to the original circuit. LUT-based primitives replace a portion of a circuit, hence they can only be layered on top of other primitives. For brevity, we denote by $M_q(s,q)$, $M_r(s,r)$, and $M_o(s,o)$ the mappings used to compute the security, risk, and overhead metrics, respectively, for an aggregation of primitives. We also use p as a compact notation for the set $\{p_{ij}|i \in \{1,...,|\mathcal{L}|\}, j \in \{1,...,n\}\}$, and adopt the same convention for q, r, and o. The actual expressions depend, however, on the specific primitives involved in the design. Risk metrics, for example, compose by taking the product of the single contributions, under the assumptions that they are independent. Therefore, by considering the logarithm of each risk variable r_{ij}, $\rho_{ij} = \log_2(r_{ij})$, the following summation rule holds:

$$\log_2(M_r(s,r)) = \sum_{i=1}^{|\mathcal{L}|}\sum_{j=1}^{n} s_{ij}\rho_{ij}. \tag{2}$$

A conservative estimate of the SAT-attack resilience takes, instead, the maximum over all the instantiated primitives, i.e.,

$$M_q(s,q) = \max_{i=1}^{|\mathcal{L}|}\max_{j=1}^{n} s_{ij}q_{ij}. \tag{3}$$

The products between binary and continuous variables in (2) and (3) can be expanded into mixed integer linear constraints via additional auxiliary variables.

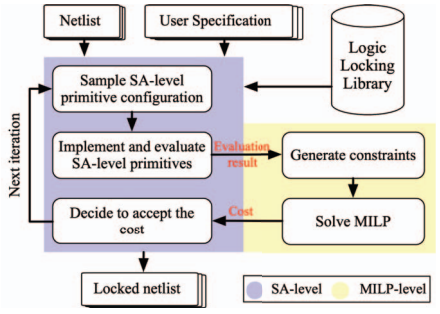


Fig. 3. Two-level optimization-based methodology.

Given the above library, composition rules, and a set of system-level requirements in terms of minimum accepted security level q_l and maximum risk r_u, we can then formulate our design problem as the following optimization problem:

$$\begin{aligned} &\underset{s,p,q,r,o}{\text{minimize}} \quad C(s,p,q,r,o) \\ &\text{s.t.} \begin{cases} M_q(s,q) \geq q_l,\ M_r(s,r) \leq r_u \\ p_{lij} \leq p_{ij} \leq p_{uij}, \forall i \in \{1,...,|\mathcal{L}|\}, \forall j \in \{1,...,n\} \\ q_{ij} = m_{qi}(p_{ij}), \forall i \in \{1,...,|\mathcal{L}|\}, \forall j \in \{1,...,n\} \\ r_{ij} = m_{ri}(p_{ij}), \forall i \in \{1,...,|\mathcal{L}|\}, \forall j \in \{1,...,n\} \\ o_{ij} = m_{oi}(p_{ij}), \forall i \in \{1,...,|\mathcal{L}|\}, \forall j \in \{1,...,n\} \end{cases} \end{aligned} \tag{4}$$

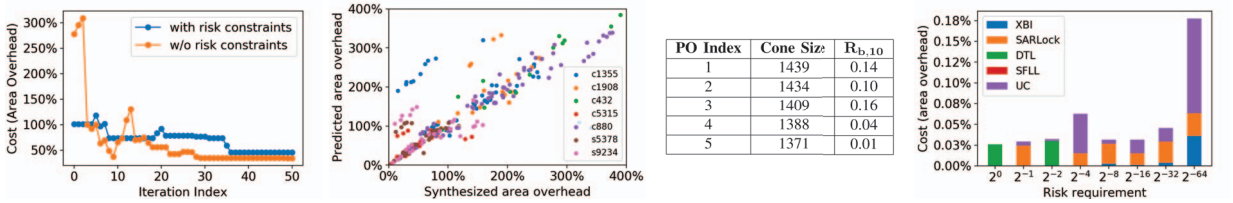
The cost C is defined by the user and, in this paper, it is the total area overhead. If the mappings m_q, m_r, and m_o are non-linear, they are approximated by piece-wise linear functions and encoded as sets of mixed integer linear constraints.

Two-Level Optimization-Based Algorithm. A compact expression for the mapping from p to q, r, and o for an aggregation of primitives may not always be available (or tractable) in closed form, and would be better computed by simulation. Therefore, we adopt a two-level optimization-based methodology to solve (4), leveraging MILP in a loop with an SA routine, as shown in Fig. 3. Our algorithm receives as inputs the original netlist and the user specification, and returns the netlist locked with an optimal construction. The LL library stores compact models for mapping primitive configurations to security, risk, and overhead metrics.

The outer optimization loop is driven by the SA routine and includes three major steps, that is, variable sampling, cost function evaluation, and cost acceptance. These are repeated until the maximum iteration count is achieved. When it is not feasible to compute the metrics in closed form for a set of primitives, we let the SA routine sample a configuration vector $\hat{p}^{SA}$ for those primitives, which is then implemented to evaluate the current security level, risk, and overhead. Both $\hat{p}^{SA}$ and its performance vector are then passed to the inner MILP solver, which instantiates a version of (4) to account for the effect of $\hat{p}^{SA}$, and solves it to obtain a new cost estimate. A cost acceptance function in the SA routine determines whether to accept this cost. If the MILP problem is, instead, infeasible, a very large cost is returned to discourage the SA routine from accepting that configuration. Finally, when the optimization terminates, the netlist will be locked based on the configuration parameter p with the lowest accepted cost.

V. Optimization Results

The optimization runs were executed on a 2.1-GHz processor with 500-GB memory. The algorithm in Sec. IV is implemented in Python and uses CPLEX [18] to solve the MILP



(a) (b)

PO Index	Cone Size	$R_{b,10}$
1	1439	0.14
2	1434	0.10
3	1409	0.16
4	1388	0.04
5	1371	0.01

(c)

(d)

Fig. 4. (a) Optimization traces for two specification sets on c5315. (b) Predicted versus simulated area overhead. (c) Learnability risk for the five largest IIR logic cones. (d) Primitives used for optimal IIR core protection.

TABLE I
SPECIFICATION OPTIONS FOR SECURITY AND RISK

Security/Risk	$E_{FC} \geq$	$t_{SAT} \geq$	$R_i \leq$	$R_r \leq$	$R_{id} \leq$
High Threshold	0.5	10^7	∞	∞	∞
Low Threshold	0.1	10^3	2^{-64}	2^{-64}	2^{-64}

problem. The LL library includes XBI [2], point-function-based techniques (SARLock [3], DTL [5], and SFLL-HD [6]), and LUT-based techniques (universal circuit (UC) [10]). The configuration for XBI and UC is determined by the SA routine, while the others are decided via MILP. Area overhead models were obtained by fitting overhead data from more than $20,000$ synthesized configurations out of the available LL primitives. The designs were synthesized using a 45-nm Nangate Open Cell Library [19].

Validation of the Methodology. 7 ISCAS benchmark circuits are selected to validate our optimization methodology. We set the maximum iteration count of the SA loop to 50 and use an exponential multiplicative cooling schedule [12]. For each circuit, we generate 32 different user specifications by permutation of the values in Table I and run our algorithm on the largest logic cone of the circuit. Fig. 4(a) shows the evolution of the cost for c5315 when we require that $E_{FC} \geq 0.5$, $t_{SAT} \geq 10^7$ with (blue) and without (orange) the constraint that R_i, R_r, R_{id} be $\leq 2^{-64}$. Both the costs settle to a significantly lower value in 40 iterations. We validate the achieved security levels by estimating E_{FC} via simulations and t_{SAT} via the execution of the SAT attack [20] with a one-day timeout time. Simulation results show that only 3.6% of the locked netlists fail to meet the E_{FC} requirement. The violation is observed when a small key-size DTL primitive is selected, due to the fact that the model adopted is less accurate in this scenario. Most of the designs reach the timeout of the SAT attack, whose duration is accurately predicted by our models [9]. Fig. 4(b) compares the actual area overhead with our prediction, showing a relative prediction error smaller than 0.5 in 87.5% of the cases. The execution time of an optimization run is dominated by the MILP portion, which takes at most 8 minutes. Overall, our approach allows effective exploration of large design spaces in terms of functional corruptibility and attack resilience.

Locking the Common Evaluation Platform. We illustrate the effectiveness of our approach on Common Evaluation Platform (CEP) [14], designed for testing different hardware security primitives. Specifically, we focus on the IIR filter core, with more than $16,000$ gates, 2724 primary input, and 2673 primary output ports, including the output and input ports of the flip-flops, respectively. Among the five largest logic cones in the circuit, we select the third cone, which has the highest Boolean learnability risk $R_{b,10}$ (with $s = 10$), as reported in Fig. 4(c). We run our algorithm with high security requirements ($E_{FC} \geq 0.5$ and $t_{SAT} \geq 2^{90}$) and 8 different risk requirements, ranging from no risk constraint to a tight bound of 2^{-64}.

Fig. 4(d) shows the locking primitives used to achieve optimal protection for different risk requirements. When no risk constraints are imposed, the optimal construction (leftmost bar) consists of two DTL blocks with key-size of 2 and 92, respectively. We find the smaller DTL block contributes to the high E_{FC} while the larger one provides the required t_{SAT}. This demonstrates the ability of our approach to explore the space and find new LL constructions via combination of primitives capable of providing security levels that are not achievable with a single existing technique. As we move toward the right of Fig. 4(d), the risk decreases significantly at a large area overhead. The increase in area is due to the partial replacement of the cone with a UC structure. For example, in the tightest risk configuration, the circuit is partially replaced, including part of a 93-bit key SARLock block, with a 7-input 5-layer universal circuit to mitigate the high isolation risk of SARLock.

VI. CONCLUSION

Our risk-aware methodology enables systematic and efficient exploration of a large range of LL design alternatives via rigorous quantification of the achievable protection levels, residual risk, and their trade-offs with overhead. By layering a set of LL primitives, our framework can also generate novel constructions that outperform existing methods used in isolation.

REFERENCES

[1] J. A. Roy *et al.*, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, 2010.
[2] J. Rajendran *et al.*, "Fault analysis-based logic encryption," *IEEE Trans. Computers*, vol. 64, no. 2, pp. 410–424, 2013.
[3] M. Yasin *et al.*, "SARLock: SAT attack resistant logic locking," in *Int. Symp. Hardware Oriented Security and Trust (HOST)*, pp. 236–241, 2016.
[4] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating sat attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, 2018.
[5] K. Shamsi *et al.*, "On the approximation resiliency of logic locking and ic camouflaging schemes," *Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 347–359, 2019.
[6] M. Yasin *et al.*, "Provably-secure logic locking: From theory to practice," in *Proc. SIGSAC Conf. Computer and Communications Security*, pp. 1601–1618, 2017.
[7] A. Sengupta *et al.*, "ATPG-based cost-effective, secure logic locking," in *IEEE VLSI Test Symp.*, pp. 1–6, 2018.
[8] H. Zhou *et al.*, "Resolving the trilemma in logic encryption.," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 918, 2019.
[9] Y. Hu *et al.*, "Security-driven metrics and models for efficient evaluation of logic encryption schemes," in *ACM-IEEE MEMOCODE*, 2019.
[10] K. Shamsi *et al.*, "On the impossibility of approximation-resilient circuit locking," in *Int. Symp. Hardware Oriented Security and Trust (HOST)*, pp. 161–170, IEEE, 2019.
[11] G. Kolhe *et al.*, "Security and complexity analysis of LUT-based obfuscation: From blueprint to reality," in *IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.
[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[13] T. Gönen and I. Ramirez-Rosado, "Review of distribution system planning models: A model for optimal multistage planning," in *IEE Proc. C (Generation, Transmission and Distribution)*, vol. 133, 1986.
[14] MIT Lincoln Laboratory, "Common Evaluation Platform," 2018.
[15] R. O'Donnell, *Analysis of Boolean functions*. Cambridge University Press, 2014.
[16] P. Chakraborty *et al.*, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation," in *Asian Hardware Oriented Security and Trust Symp. (AsianHOST)*, pp. 56–61, IEEE, 2018.
[17] J. Geist *et al.*, "RELIC-FUN: Logic identification through functional signal comparisons," in *Proc. Design Automation Conf. (DAC)*, 2020.
[18] "IBM ILOG CPLEX Optimizer."
[19] Silvaco, "45nm open cell library," 2019.
[20] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *IEEE Int. Symp. Hardware Oriented Security and Trust (HOST)*, pp. 137–143, 2015.