Closed-form Estimation of SAT-Attack Resilience

Saran Phatharodom, Nagarajan Kandasamy, and Ioannis Savidis
Department of Electrical and Computer Engineering
Drexel University, Philadelphia, PA 19104
sp694@drexel.edu, kandasamy@drexel.edu, isavidis@coe.drexel.edu

***Abstract*—A novel analytical framework to provide an estimated measure of the security of a logic-locked circuit against a Boolean satisfiability (SAT) based attack is described. Typically, a security measure based on computational complexity is estimated experimentally using time-to-solve or the number of iterative calls to the SAT solver. In this paper, a novel non-experimental approach is proposed, where a closed-form estimate of the expected computational complexity is derived. Variation in the search path lengths of an executing instance of the SAT-attack requires the development of a probabilistic model that allows for the derivation of the probability distribution as well as statistical measures of the number of iterations. The closed-form expressions for the probability distribution, the mean, and the variance of the number of iterations for SARLock, TTLock, and SFLL-flex$^{c\times k}$, all related SAT-attack resilient logic-locking techniques, are derived using the developed framework.**

***Keywords*—hardware obfuscation; logic locking; SAT-attack; security metrics**

I. Introduction

Over the past decade, many studies have shown the potential of logic locking techniques as a countermeasure to combat threats to the security of an integrated circuit, including intellectual property theft, overproduction, hardware Trojan insertion, and counterfeiting. One of the major vulnerabilities of logic locked combinational circuits is a search-space pruning attack that utilizes state-of-the-art SAT solvers, commonly referred to as a Boolean satisfiability (SAT) based attack [1], [2]. Therefore, when evaluating the effectiveness of a proposed logic obfuscation technique, SAT-attack resilience remains one of the key properties to prove the technique secure.

The resilience of logic obfuscating techniques to the SAT attack is typically evaluated experimentally, where a variety of benchmark circuits are locked with the proposed locking technique and a key of varying size. The experimental approach does not rigorously account for the possible variations in the number of calls, or iterations, to the SAT solver subroutine due to differences in the implementation of the SAT attack, as shown in [3]. As a result, multiple studies execute a single instance of the SAT-attack or a more limited number of iterations when evaluating a locked circuit, which misrepresents the true effectiveness of the proposed locking methodology. Only more recently, the SAT-attack was implemented stochastically in 1,000 trials for each evaluation of a locked circuit to induce exploration of the search path through the key space [3]. An accurate experimental setup, however, requires a greater number of samples of executing instances of the SAT attack on a given locked circuit topology as well as a more randomized experimental setup of the implementation of the SAT attack.

An estimate for the time-to-solve or the expected number of SAT-attack iterations as a function of the key size is useful to provide a measure of security without actually performing the attack, especially when predicting the expected security of a logic locked circuit with a large key size. An accurate prediction of the provided security, therefore, motivates the development of an *analytical* estimate of the SAT-resilience, which is the main contribution of this paper. A probabilistic framework is developed for deriving the closed-form expression of the expected number of iterations of a SAT-attack instance as an alternative to *experimental* sample mean.

II. SAT-attack computational complexity

The SAT-attack algorithm prunes the key-pattern search space through the process of elimination. Applying the same primary input pattern to two identical copies of the locked circuit, but with different key input patterns, results in differences in a subset of the primary outputs of the circuit. The applied input pattern that results in variation of at least one primary output bit is called a *distinguishing input pattern* (DIP). In each iteration, a DIP is determined through a SAT-solver subroutine call, which is an NP-complete problem. For each disparate output bit and with access to an unlocked active circuit (an oracle), an adversary checks which of the two values is correct. As a result, a subset of incorrect key patterns are identified and eliminated from the set of remaining candidates. The process repeats for a number of iterations λ, until only the correct key pattern(s) remain. The overall time-to-solve of a SAT-attack instance deobfuscating a logic-locked circuit is then approximately calculated as $\lambda \times t_{SAT}^{avg}$, where t_{SAT}^{avg} is the average execution time of each call to the SAT solver. The t_{SAT}^{avg} term depends on the locked circuit topology and various other factors that complicate the measure of the overall time-to-solve, and is, therefore, implementation specific. The additional factors include the differences in the underlying framework, algorithms, and configurations of the various SAT solvers, and the chosen representation of the Boolean formula applied to the SAT solver [4]. In contrast to t_{SAT}^{avg}, λ depends on the properties of the applied SAT-solver and the characteristics of the search space, with the latter providing a stronger influence. Consequently, λ is used as a proxy for the computational complexity of the SAT attack.

The analysis of SAT-resilience is applied to a series of related out-of-cone logic-locking techniques, where the key search space is well-organized and independent of the original circuit topology and functionality. First, the SARLock (SAT attack resilient logic locking) [5] and TTLock (tenacious and traceless logic locking) [6] techniques are analyzed, with each shown in Fig. 1. Both techniques generate a *one-point function* flip signal using a comparator. TTLock modifies the original circuit and, therefore, does not require a mask subcircuit to generate an exception case for the correct key column.

The SAT-resilience of an arbitrary circuit locked by SFLL-flex$^{c\times k}$, a stripped-functionality logic locking technique [7] is analyzed in Section IV. While closed-form expressions of

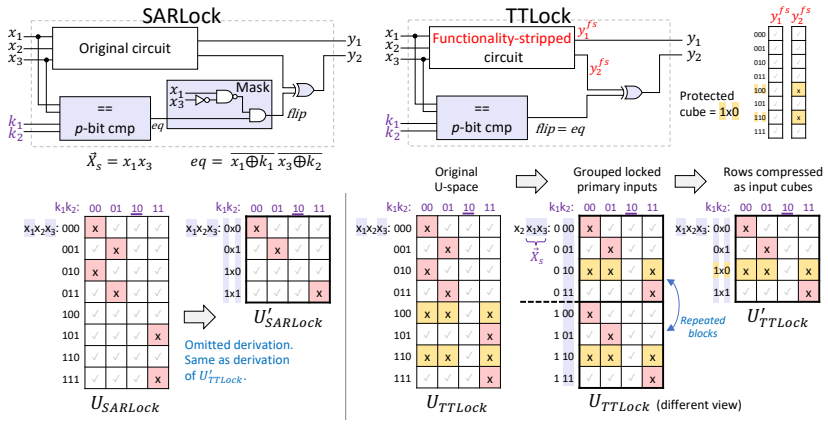


Fig. 1: An original circuit locked by SARLock and TTLock. The respective U-spaces (U) and effective compressed-cube form U-spaces (U') are also shown. The correct key vector is $\vec{K}_{correct} = k_1k_2 = 10$.

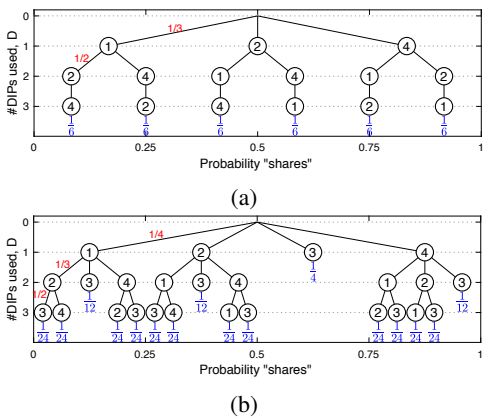


Fig. 2: Comparison of DIP-sequence trees for (a) SARLock and (b) TTLock.

the SAT-resilience of the key search space for SARLock and TTLock are manually derived, SFLL-flex$^{c\times k}$ requires a more complex analysis of the search space using computational methods.

III. Analytical Framework for Measuring Expected Computational Complexity

The steps to measure the expected computational complexity of an obfuscated circuit include (1) constructing a representation of the search space, defined as the U-space, (2) deriving a tree-model representation of the search space, and (3) computing the probability of the tree-search depth. Each step is discussed in further detail below.

A. Capturing search-space characteristics

The *U-search space*, where the "u" represents the "uncorrupted state, is a truth table with values that express whether or not the locked circuit produces a true, uncorrupted output pattern for a given primary input pattern and key pattern. The scalar truth values of 1 and 0 are denoted as ✓and ✗, respectively, to differentiate a U-truth table from the standard logical truth-table of a circuit.

For a locked circuit and an oracle, the underlying search process of an executing instance of the SAT-attack is governed by the U-search space. The structure of the values in the U-truth table characterize the search space. The U-space of a circuit locked by either SARLock or TTLock is shown in Fig. 1. Rearranging the order of the primary input variables by grouping the subset of inputs $\vec{X}_S$ that are applied to the SARLock or TTLock circuitry together as least significant bits results in a repeated pattern. Merging rows in the U-space by combining the input patterns into input cubes using X-notation (e.g. 0x0) results in the reduction (or compression) of the U-search space. Picking any DIP within the same input cube leads to the elimination of the same incorrect column(s). The computational complexity of the SAT attack is then governed by the *effective* search space U' in compressed-cube form. For ease of reference, distinguishing input cubes and U' are loosely referred to as DIP and U-space, respectively.

B. Existence of multiple search paths and DIP-sequence tree

Given the effective search space U', the set of all possible solutions of the DIP-sequences are mapped and visualized as a tree diagram, as shown in Fig. 2. The U' for both SARLock and TTLock includes four rows of input cubes. Depicted by the tree structure are all possible search paths of DIP-sequences that a SAT attack instance may follow to determine a correct key pattern. For example, referring to Fig. 2a, the left most branch of the SARLock tree is the DIP-sequence solution (DIP#1, DIP#2, DIP#3) = (1, 2, 4), where the order of the numbers represents the sequential selection of rows from the U'-search space table.

C. Probabilistic model of the search path lengths

Treating each branching decision when selecting a DIP while executing a SAT-attack as a random process, the number of iterations is consequently a random variable given as D. The DIP-sequence tree then represents the *probability space*. Without further knowledge of the specific stochastic implementation of the SAT-attack, and for simplicity of the analysis, the branching probabilities are assumed to be equally likely. For example, in Fig. 2a, the probability of a SAT attack instance deobfuscating the SARLock U-space picking the search path (1, 2, 4) = $\frac{1}{3} \times \frac{1}{2} \times \frac{1}{1} = \frac{1}{6}$. The probability mass function (PMF) of D is then the sum of the probabilities of the search paths with the same length.

D. Deriving statistical measures of the number of iterations

All DIP-sequences of the SARLock U-space are permutations of all DIPs with equal length. As a result, the PMF is a degenerate distribution with zero variance. In contrast, the DIP-sequences of the TTLock U-space are equivalent to a sampling-without-replacement problem, where there is one red ball and $2^{|\vec{K}|}-1$ white balls. The sampling game continues until either the red ball (protected cube DIP) is drawn or all the white balls (all other DIPs) are drawn. Let $\rho \equiv 2^{|\vec{K}|}$ denote the total number of balls, which is also the number of rows in the U'-search space table. The estimated PMFs and the min, max, mean, and variance of the number of iterations for SARLock and TTLock are listed in Table I and depicted in Fig. 3. The results provide an *estimate* of the number of iterations since an equally-likely branching probability is assumed.

E. Expected SAT-resilience of SARLock vs. TTLock

SARLock requires the near maximum D_{min} possible and exhibits the strongest SAT-resilience, requiring almost all

TABLE I: Estimated PMF and statistical measures of the number of iterations

	SARLock	TTLock
$\mathbb{P}(D=d)=$	$\begin{cases}1 & d=\rho-1,\\ 0 & \text{otherwise}\end{cases}$	$\begin{cases}1/\rho & d\in\{1,2,...,\rho-2\},\\ 2/\rho & d=\rho-1,\\ 0 & \text{otherwise}\end{cases}$
D_{min}	$\rho-1$	1
D_{max}	$\rho-1$	$\rho-1$
$\mathbb{E}[D]$	$\rho-1\approx 2^{\lvert\vec{K}\rvert}$	$\frac{(\rho-1)(\rho+2)}{2\rho}\approx 2^{(\lvert\vec{K}\rvert-1)}$
Var(D)	0	$\frac{\rho^2-13}{12}+\frac{2}{\rho}-\frac{1}{\rho^2}\approx\frac{2^{2\lvert\vec{K}\rvert}}{12}$

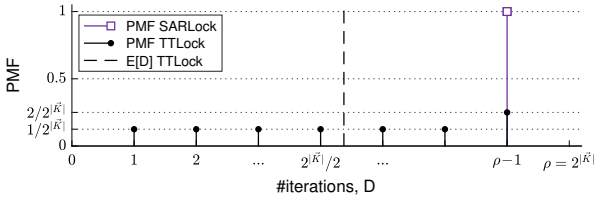


Fig. 3: PMF comparison between SARLock and TTLock for a key size of $|\vec{K}|=|\vec{X_S}|=3$.

$2^{|\vec{X}_S|}-1$ input cube patterns as DIPs. However, SARLock is vulnerable to a bypass attack [8], or majority-voting of the outputs of three copies of the locked circuit using any three randomly picked key patterns. The expected resilience $\mathbb{E}[D]$ of TTLock is about half that of SARLock. In addition, TTLock is vulnerable to a rare and fortuitous execution of the SAT-attack, which results in a worst case D_{min} of 1. For both techniques, $\mathbb{E}[D]$ increases exponentially as the key size $|\vec{K}|$ is increased.

IV. Computational approach for complex U-spaces: A case study using SFLL-flex$^{c\times k}$

The patterns of 1s and 0s within the U-space tables of SARLock and TTLock are simple, with only one or two lines of corruption, and the corresponding DIP-sequence trees are straightforward to analyze. In this section, a U-space with a more complex pattern of 1s and 0s that results in corruption beyond a few lines is considered. Analyzing the branching pattern of the resulting DIP-sequence tree becomes much more difficult. Therefore, an automated quantitative approach is pursued in which a collection of U-spaces is simulated for various combinations of tunable parameters of the given obfuscation technique. Next, a novel algorithm is utilized to enumerate *all* SAT-attack DIP sequence solutions for each generated U-space table. Assuming an equally likely branching probability, the estimated PMF and statistical measures of the number of used DIPs are then calculated from the list of all DIP sequence solutions.

The proposed algorithm is computational limited due to the combinatorial explosion of enumerating all DIP-sequence solutions. The worst-case total number of all DIP-sequence solutions is given by $(2^{|\vec{X}_S|})!$, which corresponds to all permutations of every row in the U-space table when all input patterns are required as DIPs to eliminate all incorrect keys. Therefore, a small circuit is implemented to demonstrate the merit of the proposed algorithm, and provide the analysis and insight that can then be extrapolated to larger circuits. Even for small circuit sizes where $|\vec{X}_S|\leq 3$, the resulting analysis yields significant insight, leading to the derivation of closed-form expressions of the statistics characterizing the SAT-resilience.

At a minimum, from the computed PMF and statistical analysis of the number of iterations, a qualitative trend of SAT-resilience as a function of tunable parameters such as key size is obtained. Curve-fitting techniques are applicable for extracting *approximated* closed-form expressions from the data points. For some of the locking techniques, the observed data patterns lead to a derivation of the *exact* closed-form expressions of the expected SAT-resilience.

In this section, the more complicated U-space of the SFLL-flex$^{c\times k}$ locking technique [7] is analyzed by applying a computational approach. A well-organized behavior is observed within the U-space table, which resulted in the derivation of the exact closed-form expressions of the PMF and statistic measures of the SAT-resilience of the circuit.

A. SFLL-flex$^{c\times k}$ logic locking

Upon the introduction of the SAT-attack algorithm [1], [2] in 2015, a series of out-of-cone logic locking techniques were proposed [5]–[7], [9]. SARLock [5] provided the foundation for the development of TTLock [6], which evolved into SFLL-flex$^{c\times k}$ [7]. For SFLL-flex$^{c\times k}$, c denotes the number of protected cubes and k the bit size of each given cube. The technique allows for flexibility when choosing between any combinations of protected cubes to obfuscate the circuit. A lookup table (LUT) representation of SFLL-flex$^{c\times k}$ is shown in Fig. 4a. To more clearly illustrate the locking mechanism of SFLL-flex$^{c\times k}$, an equivalent-circuit topology is provided that implements SFLL-flex$^{c\times k}$ using multiple comparators as shown in Fig. 4b. Instead of using one comparator to lock a protected cube, which is the case with TTLock, SFLL-flex$^{c\times k}$ uses multiple comparators to generate flip signals that are then OR-ed together. Each comparator checks if the current values of the subset of the input ("sub-input") pattern $\vec{X}_S$ matches with the secret "sub-key" of that comparator. Therefore, SFLL-flex$^{c\times k}$ provides a flexible means to set any number of protected cubes.

A flip vector describes which of the primary outputs are to be flipped when a sub-input pattern $\vec{X}_S$ matches a sub-key pattern. If the sub-key pattern is correct, the pre-flipped functionally-stripped outputs are restored. Otherwise, the un-stripped outputs that were initially correct are corrupted by the flip function instead. In the implementation of SFLL-flex$^{c\times k}$ shown in Fig. 4b, the flip vectors are optionally treated as an additional set of secret keys, which expand the U-search space further. If a flip vector with a value of 1 is implemented by directly connecting the output of the comparator to the input of the OR gate of the flip-function, thereby bypassing the AND gate, the flip vector bit is fully observable and does not act as a secret key. Similarly, a flip vector with a value of 0 is implemented by simply not connecting the comparator output to the specific OR gate of the flip-function.

B. Effect of design choices of SFLL-flex$^{c\times k}$ on SAT-resilience

The factors that affect the SAT-resilience of a circuit locked by the SFLL-flex$^{c\times k}$ technique are described below.

1) Tunable design parameters: The locked circuit performance and overhead depend on the size of the LUT given by the relationship $c(k+f)$, where c is the number of protected cubes, k is the bit-size of each protected cube, and f is the bit-size of each flip vector (number of flipped outputs) [7].

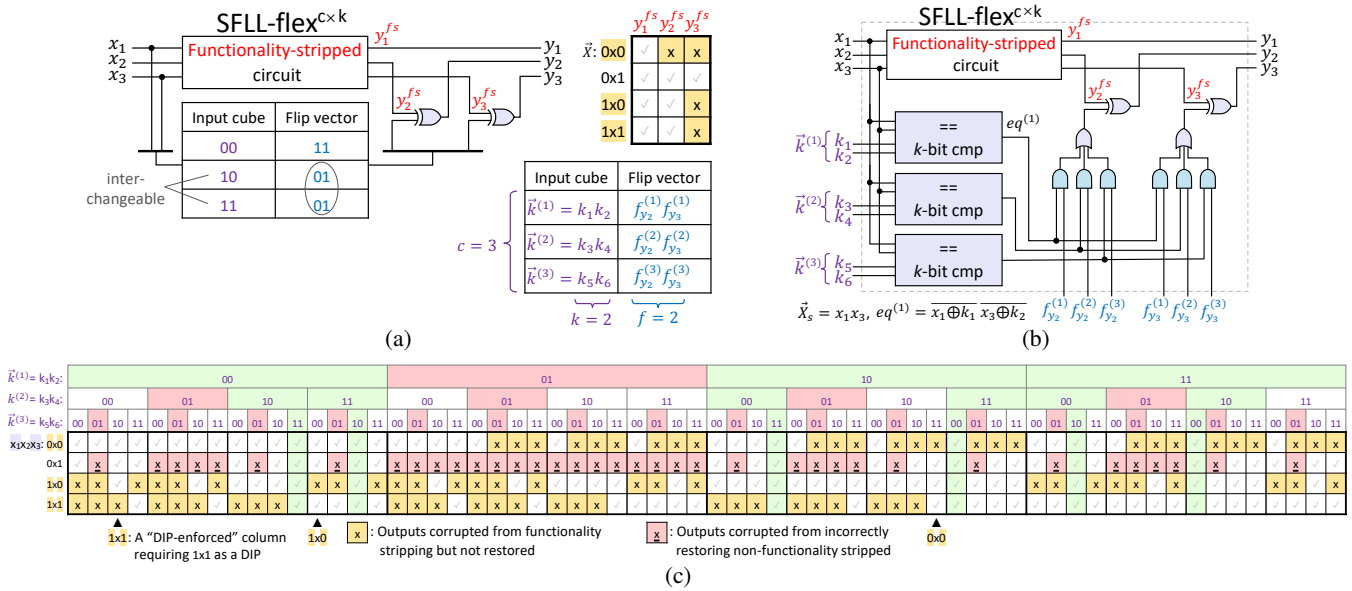


Fig. 4: A case study of applying SFLL-flex$^{c\times k}$ on a circuit that includes (a) the original LUT representation of the circuit [7], (b) an implementation of SFLL-flex$^{c\times k}$ on an arbitrary circuit based on multiple comparators and a flip vector configuring inputs as optional additional secret keys, and (c) the corresponding U-space, where $c \times k = 3 \times 2$ and all flip vectors are the same and fully observable, which implies all sub-key patterns are interchangeable. There are multiple vectors of correct keys $\vec{K}_{correct}$ such as (00, 10, 11) and (00, 11, 10).

To simplify the analysis, the flip vectors are assumed to be fully observable and not part of the secret keys. The strength in security of the obfuscation technique is characterized by c, k, and f. The correct value of the k-bit sub-key is the secret protected cube pattern. The value of k is also determined by $|\vec{X}_S|$, which is the number of primary inputs provided to the comparators of the subcircuit generating the flip signal. An example of an arbitrary 3-input by 3-output locked circuit is shown in Fig. 4a, where $c = 3$, $|\vec{X}_S| = k = 2$, and $f = 2$.

2) Duplicate flip vector patterns: Within the LUT, the rows are not order-sensitive. Swapping an entire row of the LUT with another row yields the same U-space. If the flip vectors are fully observable, each protected cube pattern forms a pair with a corresponding flip vector pattern. If, however, the flip vectors include duplicates, the sub-key patterns are interchangeable within the rows of the LUT with other duplicate flip vectors, as shown in Fig. 4a. The analysis of the SAT-resilience, however, becomes more complex as the corresponding U-space patterns differ depending on the duplicity of the flip vector patterns. The simple case when only one primary output is corrupted ($f = 1$) is analyzed in this paper. The analysis generalizes to cases in which multiple outputs are corrupted ($f > 1$) but all flip vector patterns are exactly the same.

3) Duplicate sub-key patterns: U-spaces that include duplicate sub-key patterns, or equivalently, duplicate protected cubes, are shown to produce a different SAT-resilience trend as compared to U-spaces that include only unique sub-key patterns. As selecting duplicate sub-keys is possible, when evaluating the SAT-resilience of a circuit, patterns due to duplicated sub-keys must be included in the search pool. As a result, the original SAT attack described in [1], [2] does not require a modification to reduce the key search space to include only the combinations of unique sub-keys.

4) Protected cube patterns: The set of selected protected cube patterns does not affect the expected SAT-resilience. For example, in the U-space shown in Fig. 4c, choosing any three rows of the four as the protected cubes yields a U-space with equal expected SAT-resilience. Holding all other factors constant, changing the specific values of the protected cube patterns results in an equivalent shaped DIP-sequence tree with different node labels. Consequently, the number of DIPS required is unchanged.

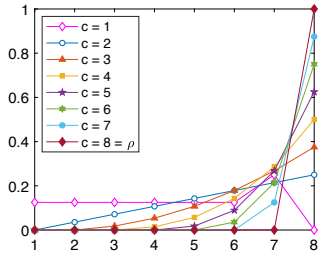
In summary, the following are assumed for the analysis of SAT-resilience, which covers a specific configuration of SFLL-flex$^{c\times k}$, while is also in line with the experimental evaluation in the original paper [7].

- Flip vector patterns are fully observable and are not part of the secret keys.
- Flip vector patterns are the same for every protected cube.
- All sub-key patterns are unique.
- The original SAT attack algorithm described in [1], [2] is utilized, while assuming an equally-likely branching probability.

C. Visual characterization of the representative U-space

The objective is to gain insight on the U-space of a small locked circuit, which can then be generalized to larger circuits and any combination of tunable parameters. The approach is to start with a small circuit, which generates a U-space that is more easily visualized.

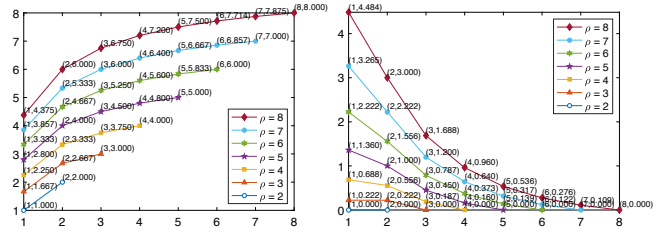
MATLAB code is used to generate U-space matrices of 1s and 0s for combinations of small k and c values. The bit size of each cube k is initially set to 2 and 3, while c is set to vary from 1 to 2^k. When c is 1, SFLL-flex$^{c\times k}$ behaves similar to TTLock. Therefore, the U-space and security metrics are also the same. The SFLL-flex$^{c\times k}$ U-space matrix is of the size



	$d=1$	$d=2$	$d=3$	$d=4$	$d=5$	$d=6$	$d=7$	$d=8$
$c=1$	$1/8$	$1/8$	$1/8$	$1/8$	$1/8$	$1/8$	$2/8$	$0/8$
$c=2$	0	$1/28$	$2/28$	$3/28$	$4/28$	$5/28$	$6/28$	$7/28$
$c=3$	0	0	$1/56$	$3/56$	$6/56$	$10/56$	$15/56$	$21/56$
$c=4$	0	0	0	$1/70$	$4/70$	$10/70$	$20/70$	$35/70$
$c=5$	0	0	0	0	$1/56$	$5/56$	$15/56$	$35/56$
$c=6$	0	0	0	0	0	$1/28$	$6/28$	$21/28$
$c=7$	0	0	0	0	0	0	$1/8$	$7/8$
$c=8$	0	0	0	0	0	0	0	$1/1$

(a) (b)

Fig. 5: An analysis of SFLL-flex$^{c\times k}$ U-spaces with results shown for (a) PMFs $\mathbb{P}(D=d)$ as a function of D and (b) the tabulated PMF for each row. The set parameters include the sub-input vector $|\vec{X}_S| = k = 3$, the number of rows $\rho = 2^k = 8$, and for an increase in the number of cubes c from 1 to ρ, assuming an equally-likely branching probability.



(a) (b)

Fig. 6: Characterization of the (a) means $\mathbb{E}[D]$ and (b) variances $\mathrm{Var}(D)$ of the PMFs of the simulated SFLL-flex$^{c\times k}$ U-spaces, both as a function of c and for $\rho = 2$ to 8. An equally-likely branching probability is assumed.

$\rho \times \rho^c$, where $\rho = 2^{|\vec{X}_S|}$ is the number of rows in the U-space table.

The U-space generated when $c = 3$, and $k = 2$ is shown in Fig. 4c. Note that two types of output corruptions are possible. The first is when the output is functionally stripped but is not restored, which occurs at the rows of the U-space that are protected cubes. The second type of corruption is when the output is not stripped but is incorrectly restored, which occurs at rows of non-protected cubes. There are multiple ($c!$) correct key patterns as the order of the sub-keys does not matter when the flip vector patterns are the same for every protected cube.

In addition, there exist multiple "DIP-enforced" columns that require each protected cube as a DIP included within any SAT-attack search path. For the SAT attack to terminate, every incorrect column must be eliminated. If a column has a corrupt output for only one input cube pattern, that pattern must then be selected as a DIP. A "DIP-enforced" column occurs when $c-1$ sub-keys are different and correct, but one remaining sub-key is a duplicate of the other correct sub-key. Consequently, the lower bound $D_{min} \geq c$ is implied. Therefore, *all* protected cubes are always necessary, unlike the TTLock U-space, where the SAT attack is able to resolve the correct key if all non-protected cubes are used, whether or not the one protected cube is used.

The upper bound of D_{min} is c. Each incorrect column is missing at least one correct sub-key pattern. For the example showin in Fig. 4c, each incorrect column must contain at least one yellow x-cell representing an unrestored corrupted output pattern. Therefore, utilizing all c protected cube DIPs

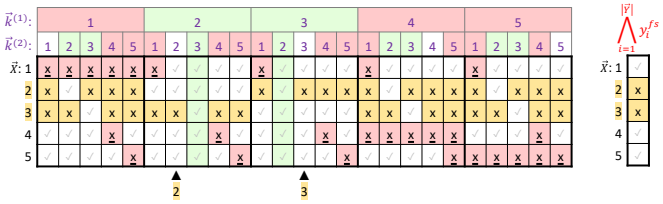


Fig. 7: A U-space of an abstract circuit locked with SFLL-flex$^{c\times k}$, where the number of rows $\rho = 2^k$ is generalized to a non-power of 2. For the given circuit, $\rho = 5$ and $c = 2$. The input patterns and key patterns are generalized to the integers 1, 2, 3, ... , ρ.

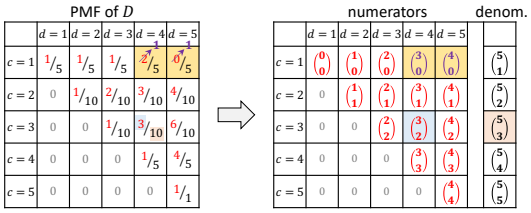


Fig. 8: Simulated PMFs expressed in terms of binomial coefficients for $\rho = 5$.

is enough to remove all incorrect columns, validating the upper bound $D_{min} \leq c$. As $D_{min} \geq c$ and $D_{min} \leq c$, $D_{min} = c$ is implied.

D. Results from numerical analysis

To simulate the equally-likely DIP selection behavior of a SAT-attack instance, an algorithm developed in MATLAB is used to iterate through all valid DIP choices at each branching point. The algorithm operates on the U-space matrix in a tree traversal manner in order to enumerate all DIP sequence solutions. The accumulated constraints from eliminated candidate key columns by using past DIPs are captured by keeping track of the status of the remaining candidate columns. The list of all DIP sequence solutions is then used to compute the PMF and statistical measures of D, assuming each branching is equally likely. No SAT-solvers are used in the analysis. The exponential increase in the number of all DIP sequences as the sub-inputs increase requires the study of a relatively small U-space. In this paper, 3-bit sub-input ($|\vec{X}_S| = k = 3$) is suffice for the analysis to be useful, while is fast enough to compute.

The PMFs as a function of D and as c increases from 1 to ρ, for $\rho = 2^{|\vec{X}_S|} = 2^3 = 8$, is shown in Fig. 5. As c increases, the curve becomes exponential. The relationship of $D_{min} = c$ is observed as predicted. Note that when $c = 1$ the PMF becomes that of the TTLock U-space. For $c > 1$, a well organized pattern is observed. Referring to Fig. 5b, when observing the values along each column, the numerators follow a pattern of a Pascal triangle. For the $d = 8$ column, the denominators also follow the pattern of a Pascal triangle. The mean and the variance both as a function of the number of protected cubes c for a ρ of 2^2 and 2^3 are shown in Fig.6.

To observe trends in the mean and variance, more values of ρ are needed. To avoid increasing the simulation time, non-physical values (non-power of two) of the number of rows are considered. An example abstract U-space with $\rho = 5$ is

TABLE II: Estimated PMF and statistical measures of the number of iterations

SFLL-flex$^{c\times k}$,	unique sub-keys, repeated flip vectors, $1 < c \leq \rho$
$\mathbb{P}(D=d) =$	$\begin{cases} \frac{\binom{d-1}{c-1}}{\binom{\rho}{c}} & c \leq d \leq \rho, d \in \mathbb{N}, \\ 0 & \text{otherwise} \end{cases}$
D_{min}	c
D_{max}	ρ
$\mathbb{E}[D]$	$(\rho+1)\frac{c}{c+1} = (2^k+1)\frac{c}{c+1}$
$\mathbb{E}[D^2]$	$\rho^2 - (\frac{\rho-c}{c+1})(\frac{2\rho c+2\rho+c}{c+2})$
Var(D)	$\mathbb{E}[D^2] - \mathbb{E}[D]^2 = (\frac{\rho-c}{c+1})(\frac{c}{c+1})(\frac{\rho+1}{c+2})$

shown in Fig. 7. The abstract U-space is of the size $\rho \times \rho^c$. With more values of ρ, the plotted mean and variance exhibit observable patterns. The results shown in Fig. 6 indicate that as c increases, $\mathbb{E}[D]$ increases but at diminishing rate toward the maximum possible value of $D = \rho$. In contrast, Var(D) decreases at a faster rate toward zero as c increases. When $c = \rho$, the expected SAT-resilience is the degenerate PMF similar to that of the SARLock U-space.

At this stage, curve fitting techniques are applicable to extract expressions for approximations of the statistical measures of the SAT-resilience, as a function of the tunable obfuscation parameters c and k. However, as a Pascal triangle pattern was observed in the PMF, closed-form statistical expressions of the $\mathbb{E}[D]$ and Var(D) are derived.

First, due to the Pascal triangle pattern, the PMFs are expressed in terms of binomial coefficients. The numerator is given by $\binom{d-1}{c-1}$ and the denominator by $\sum_{d=c}^{\rho}\binom{d-1}{c-1} = \binom{\rho}{c}$, after applying the hockey-stick identity. Except in the case when $c = 1$, the PMF is given by $\binom{d-1}{c-1}/\binom{\rho}{c}$, $d \in \{c, c+1, ..., \rho\}$.

Second, a certain linearity in ρ is observed in the characterization of the expected value of D as a function of c, which is shown in Fig. 6a. For each increment in ρ, $\mathbb{E}[D]$ increases by a fixed amount of $c/(c+1)$. Therefore, $\mathbb{E}[D] = c+(\rho-c)\frac{c}{c+1}$ is directly observed.

The final step is to determine Var(D). The previous insight of TTLock being analogous to ball sampling without replacement and that every DIP sequence solution for $c > 1$ must include all protected cubes indicates that the characteristics of the underlying PMF also follow a similar sampling without replacement, where the number of red/special balls is c out of ρ total number of balls. The probability of using d number of draws until retrieving all c red balls matches the observed PMF expression.

$$\mathbb{P}(\text{uses } d \text{ draws}) = \mathbb{P}(\text{draws } c-1 \text{ red balls in } d-1 \text{ draws}) \times \mathbb{P}(\text{draws the last red ball}) \quad (1)$$

The expressions for $\mathbb{E}[D]$ and $\mathbb{E}[D^2]$ are derived directly from the expression of the PMF using the hockey-stick identity, which results in the expression of Var(D).

The expressions for the PMF of D and the corresponding statistical measures for the SFLL-flex$^{c\times k}$ U-space, as functions of c and $\rho = 2^k$ are summarized in Table II. When c is 1, the previously derived expressions for TTLock are used.

E. Effects of c and k on the expected SAT-resilience

From the closed-form expressions listed in Table II, the expected number of DIPs required grows exponentially with an increase in k, while growing more slowly with increases in c. If the total number of key bits $c * k$ is fixed, the closed-form solution indicates that configuring $c = 1$ yields a higher $\mathbb{E}[D]$ though with the worst D_{min}. Increasing c results in an increase in D_{min}. However, the much greater drop in $\mathbb{E}[D]$ outweighs the benefit.

The derived close-form expressions indicate that k must be as large as possible, though $k = |\vec{X}_s| \leq |\vec{X}|$ is limited by the number of primary inputs. Then, the number of protected cubes c should be increased, though the marginal gain in the expected SAT-resilience per increment of c diminishes. On the other hand, with respect to circuit area overhead, c and k should be kept on the lower spectrum.

V. Conclusions

A probabilistic model of the lengths of the search paths taken by a SAT attack instance deobfuscating a logic locked circuit is provided. The model allows for the derivation of closed-form expressions of the estimates of the PMF and statistical properties of the number of iterations expressed as functions of the key size or locking-scheme tune-able parameters. The statistical framework provides an efficient means to quantify the expected SAT-resilience based on the assumption of an equally-likely branching probability when selecting a DIP. The derived expressions of the SAT-resilience are useful when predicting the security of a circuit that includes a large number of key bits and to also compare between locking techniques.

The analysis provided indicates that for the same key size, the expected SAT-resilience of TTLock is about half of that of SARLock. For SFLL-flex$^{c\times k}$, as the number of cubes increases, the expected SAT-resilience increases at a diminishing rate. For the same total key size, SARLock provides greater SAT-resilience than SFLL-flex$^{c\times k}$.

References

[1] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137–143, May 2015.

[2] M. El Massad, S. Garg, and M. V. Tripunitara, "Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes.," *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pp. 1–14, Feb. 2015.

[3] K. Juretus and I. Savidis, "Importance of Multi-parameter SAT Attack Exploration for Integrated Circuit Security," *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 366–369, Oct. 2018.

[4] A. Biere and M. J. Heule, "The Effect of Scrambling CNFs," *Proceedings of Pragmatics of SAT*, Vol. 59, pp. 111–126, Jul. 2019.

[5] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236–241, May 2016.

[6] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, "What to Lock?: Functional and Parametric Locking," *Proceedings of the Great Lakes Symposium on VLSI*, pp. 351–356, May 2017.

[7] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice," *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1601–1618, Oct. 2017.

[8] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against All Known Logic Locking Attacks," *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 189–210, Sept. 2017.

[9] 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, Feb. 2019.