Efficacy of Satisfiability-Based Attacks in the Presence of Circuit Reverse-Engineering Errors

Qinhan Tan
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
tanqinhan@zju.edu.cn

Seetal Potluri
Department of ECE
North Carolina State University
Raleigh, U.S.
spotlur2@ncsu.edu

Aydin Aysu
Department of ECE
North Carolina State University
Raleigh, U.S.
aaysu@ncsu.edu

***Abstract*—Intellectual Property (IP) theft is a serious concern for the integrated circuit (IC) industry. To address this concern, logic locking countermeasure transforms a logic circuit to a different one to obfuscate its inner details. The transformation caused by obfuscation is reversed only upon application of the programmed secret key, thus preserving the circuit's original function. This technique is known to be vulnerable to Satisfiability (SAT)-based attacks. But in order to succeed, SAT-based attacks implicitly assume a *perfectly* reverse-engineered circuit, which is difficult to achieve in practice due to reverse engineering (RE) errors caused by automated circuit extraction. In this paper, we analyze the effects of random circuit RE-errors on the success of SAT-based attacks. Empirical evaluation on ISCAS, MCNC benchmarks as well as a fully-fledged `RISC-V CPU` reveals that the attack success degrades exponentially with increase in the number of random RE-errors. Therefore, the adversaries either have to equip RE-tools with near perfection or propose better SAT-based attacks that can work with RE-imperfections.**

***Index Terms*—IP-Theft, Logic-Locking, Satisfiability-Based Attacks, Reverse-Engineering Errors**

I. Introduction

Logic locking has been proposed to mitigate hardware IP-theft, which is a serious concern for the IC industry [1]–[18]. Logic locking is, unfortunately, known to be vulnerable to Satisfiability (SAT)-based attacks [19]–[29]. But all SAT-based attacks assume a *perfectly* reverse-engineered circuit. This assumption may be invalid if the adversary purchases the IC from the open market, captures images and uses machine learning for automated circuit extraction, which is error prone [30]–[32]. There is, however, no prior work on evaluating the impact of reverse-engineering (RE) errors on attack accuracy. In this paper, for the first time, we analyze and quantify the effect of such errors. The primary contributions of the paper are as follows:

- We propose an error model for RE and evaluate the attack accuracy for single as well as multiple random error scenarios.
- We identify different error scenarios which can and cannot be canceled out by applying the suitable key.
- Using benchmarks as well as a `RISC-V CPU`, we quantify the impact of RE-errors on the attack accuracy.
- Surprisingly, we unveil that in the presence of errors, increasing logic locking amount can reduce the security.

Our research reveals that SAT-attacks can fail even if there is a single RE-error, but there may be cases of attack success with a reasonably high error count. The conclusions of our paper is threefold. First, the RE tools typically need high accuracy for attacking logic locking. Second, end-user adversary needs better SAT attacks to work with RE-errors. Third, in majority of the cases adding more locks to the circuit increases attack success (reduces the security-level) in the presence of errors.

II. Satisfiability-based attack

SAT-solvers find the inputs to a Boolean expression such that the expression evaluates to true. The adversary can thus formulate the locked circuit obtained through RE as a Boolean expression and use Boolean SAT-solvers to find a secret key for the given input-output pairs. The fundamental challenge is, however, to extract the secret key by observing a limited set of input-output pairs. Therefore, straightforward SAT-solver usage is ineffective—although it returns a key, the key may only satisfy the limited set and not guaranteed to satisfy input-output pairs outside this set. SAT-attack solves this problem by two intuitions.

First, unlike cryptographic constructions, the secret key itself may not be unique, causing multiple keys to form equivalence-classes that would generate identical input-output pairs. Second, a wrong-key equivalence-class can be pruned off by determining the distinguishing input pattern (DIP), a certain input pattern which produces different outputs for two keys from two different equivalence classes. As a result, the attack aims to perform a brute-force search on key equivalence-classes, which is more efficient than performing brute-force search on individual keys.

The attack at a high-level works as follows. The adversary reconstructs the locked netlist through RE and formulates the circuit as a SAT-instance to find the DIPs. The adversary then gets an activated chip from the market that can generate valid input-output pairs. This chip is referred to as the oracle. The adversary applies the DIP to the oracle and records the output. The DIP-output pair is used to prune-off wrong equivalence class(es) of keys by updating the SAT-instance. The process continues until there is only one equivalence class left, which has to be the equivalence class of the correct keys.

Now we formally describe the attack. Let the reverse-engineered locked circuit be $C(\vec{X}, \vec{K}, \vec{Y})$, which has input vector $\vec{X}$ and output vector $\vec{Y}$, and which is locked with key

Algorithm 1 SAT-attack Logic Decryption Algorithm [19]

Function: *decrypt*.
Inputs: C and *eval*.
Output: $\vec{K_c}$

1: $i := 1$
2: $F_1 = C(\vec{X}, \vec{K_1}, \vec{Y_1}) \wedge C(\vec{X}, \vec{K_2}, \vec{Y_2})$
3: **while** $sat[F_i \wedge (\vec{Y_1} \neq \vec{Y_2})]$ **do**
4: $\quad \vec{X_i^d} := sat\_assignment_{\vec{X}}[F_i \wedge (\vec{Y_1} \neq \vec{Y_2})]$
5: $\quad \vec{Y_i^d} := eval(X_i^d)$
6: $\quad F_{i+1} = F_i \wedge C(\vec{X_i^d}, \vec{K_1}, \vec{Y_i^d}) \wedge C(\vec{X_i^d}, \vec{K_2}, \vec{Y_i^d})$
7: $\quad i := i + 1$
8: **end while**
9: $\vec{K_c} := sat\_assignment_{\vec{K_1}}(F_i)$

vector $\vec{K}$. Let the locked circuit has M input bits and L gates. Algorithm 1 shows the SAT-attack in detail, where i signifies the iteration number. F_1 is the initial SAT-instance and F_i is the SAT-instance in the i^{th} iteration. Each step of the algorithm is defined as follows:

Step 2: Formulates the SAT-instance as two copies of the locked circuit $C(\vec{X}, \vec{K_1}, \vec{Y_1})$ and $C(\vec{X}, \vec{K_2}, \vec{Y_2})$ with same input $\vec{X}$ but different keys $\vec{K_1}$, $\vec{K_2}$ and different outputs $\vec{Y_1}$, $\vec{Y_2}$. In the next step, this formulation is exploited to generate different keys that produce different outputs.

Step 3: Checks if the there are at least two different equivalence classes that satisfy the current SAT-instance. If this condition is satisfied, it enters the loop, otherwise terminates the loop;

Step 4: Runs the SAT-solver on the current SAT-instance. From the returned SAT assignment $\{\vec{X}, \vec{K_1}, \vec{Y_1}, \vec{K_2}, \vec{Y_2}\}$, this step also extracts the DIP for i^{th} iteration, denoted as $\vec{X_i^d}$,

Step 5: Evaluates the oracle output for $\vec{X_i^d}$, denoted as $\vec{Y_i^d}$.

Step 6: Adds corresponding DIP-output constraints to the SAT-instance ($C(\vec{X_i^d}, \vec{K_1}, \vec{Y_i^d}) \wedge C(\vec{X_i^d}, \vec{K_2}, \vec{Y_i^d})$) to eliminate wrong-key equivalence-class(es) for the i^{th} iteration.

Step 9: Executes the SAT-solver on the SAT-instance after loop termination and extracts $\vec{K_1}$ from the returned SAT-assignment $\{\vec{X}, \vec{K_1}, \vec{Y_1}, \vec{K_2}, \vec{Y_2}\}$. Since the while condition in line 3 fails if and only if there is a single equivalence class left, $\vec{K_1}$ is guaranteed to be *functionally correct*, i.e., $\vec{K_c} = \vec{K_1}$. This is a unique property of SAT-attack, which preempts verification. Otherwise even if the attacker finds the correct key, the verification step needs checking outputs for all possible 2^M input patterns, which is computationally infeasible for large circuits.

III. Analyzing the Impact of RE-Errors

The errors in the circuit RE can cause decryption to return the wrong keys, that is otherwise guaranteed to return the correct ones. This section analyzes the effects of the RE errors by describing the SAT-instance formulation by using a toy example.

SAT-solvers require inputs organized in a special format, called the conjunctive normal form (CNF), such that the expression evaluates to $true$. Table I shows an example for the CNF formulation of or, and, $nand$, and xor gates. Figure 1 (a) shows a 2-bit input 1-bit output sample circuit. This Boolean function takes a, b as inputs and computes $c = \overline{a.b}$. Figure 1 (b) shows the corresponding locked version with random insertion [1] of 1 xor-type key-gate (shown in red). Note that the *correct key* in this case is '1'. Equations 1 and 2 show the conversion of the locked circuit to SAT-instances $C(\vec{X}, \vec{K_1}, \vec{Y_1})$ and $C(\vec{X}, \vec{K_2}, \vec{Y_2})$ respectively using Table I, where $C(.)$, the portion in black corresponds to the and gate and the portion in red corresponds to the xor key-gate.

TABLE I
Formulating Logic Gates as CNF for SAT-attack

Logic gate	Boolean clauses
$z = or(x_1, x_2)$	$(\overline{x_1} + z).(\overline{x_2} + z).(x_1 + x_2 + \overline{z})$
$z = and(x_1, x_2)$	$(x_1 + \overline{z}).(x_2 + \overline{z}).(\overline{x_1} + \overline{x_2} + z)$
$z = nand(x_1, x_2)$	$(x_1 + z).(x_2 + z).(\overline{x_1} + \overline{x_2} + \overline{z})$
$z = xor(x_1, x_2)$	$(\overline{x_1} + \overline{x_2} + \overline{z}).(x_1 + x_2 + \overline{z}).(\overline{x_1} + x_2 + z).(x_1 + \overline{x_2} + z)$

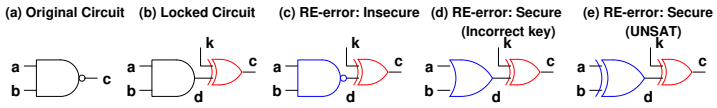

Fig. 1. Motivational example to analyze impact of RE-errors

$$C(\vec{X}, \vec{K_1}, \vec{Y_1}) = \Big[(a + \overline{d_1})(b + \overline{d_1})(\overline{a} + \overline{b} + d_1)(\overline{d_1} + \overline{k_1} + \overline{c_1})(d_1 + k_1 + \overline{c_1})(\overline{d_1} + k_1 + c_1)(d_1 + \overline{k_1} + c_1)\Big] \quad (1)$$

$$C(\vec{X}, \vec{K_2}, \vec{Y_2}) = \Big[(a + \overline{d_2})(b + \overline{d_2})(\overline{a} + \overline{b} + d_2)(\overline{d_2} + \overline{k_2} + \overline{c_2})(d_2 + k_2 + \overline{c_2})(\overline{d_2} + k_2 + c_2)(d_2 + \overline{k_2} + c_2)\Big] \quad (2)$$

Equations 1 and 2 are used to generate F_1 in step 2 of the algorithm. Subsequently, in the first iteration, step 4 of the algorithm returns $\{a, b, k_1, k_2, c_1, c_2\}$=$\{000101\}$, thus the first DIP is $\vec{X_1^d}$=$\{a, b\}$=00 and the corresponding oracle response is $\vec{Y_1^d}$=$\overline{0.0}$=1. Substituting $\vec{X_1^d}$ and $\vec{Y_1^d}$ in step 6 yields:

$F_2 = F_1 \wedge (d_1 \oplus k_1) \wedge (d_2 \oplus k_2)$ (3).

Using equation 3 in second iteration, the while loop in step 3 fails, because $F_2 \wedge (c_1 \neq c_2)$ is unsatisfiable. Thus, the loop terminates and running the SAT-solver on F_2 (step 9) gives $\vec{K_c}$=1, which makes the locked circuit's Boolean function $(a.b) \oplus \vec{K_c}$=$(a.b) \oplus 1$=$\overline{a.b}$. Thus, $\vec{K_c}$ is functionally correct, in the absence of RE-errors. The SAT-attack is able to find the correct key with just one DIP for this toy circuit.

A. RE-Error Model

In this paper, we assume only RE-errors for 2-input logic gates. Since the possibilities are xor, $xnor$, $nand$, nor, and and or, we consider the possible RE-errors as the possible interpretation errors between the candidate choices within this list. For example, a 2-input $nand$ gate within original netlist could be erroneously reverse-engineered as one of the candidates within the list $\{nor, xor, xnor, and, or\}$. If, e.g., it was erroneously reverse-engineered as and, we refer to it as $nand \rightarrow and$ type RE-error. To cover all possible cases, we also assume that the errors can occur randomly.

Even with RE-errors, the attack can return a *functionally correct key*. We term them as "RE-errors not improving the security". By contrast, for certain RE-errors (1) the attack can return a *functionally incorrect key*; or (2) the SAT-instance could be unsatisfiable (the SAT-solver returns *UNSAT*). We term them as "RE-errors improving the security". Next, we discuss each of these categories in more detail.

B. RE-errors not improving the security

Case 1: Figure 1 (c) shows a locked circuit with $and \rightarrow nand$ type of RE-error with the faulty gate shown in blue. For this case, all the steps explained in the absence of error remains the same, except that in the last step, the algorithm returns $\vec{K_c}$=0, which makes the locked circuit Boolean function $\overline{(a.b)} \oplus \vec{K_c} = \overline{(a.b)} \oplus 0 = \overline{(a.b)}$. Thus, $\vec{K_c}$ is functionally correct and the **RE-error is absorbed by the key**, thus making the attack successful despite the error. This is an example of RE-error that does not improve the security.

C. RE-errors improving the security

Case 2: Figure 1 (d) shows a locked circuit with $and \rightarrow or$ type of RE-error, with the faulty gate shown in blue. In this case,

$$F_1 = \Big[(\bar{a}+d_1)(\bar{b}+d_1)(a+b+\overline{d_1}) \,.(\overline{d_1}+\overline{k_1}+\overline{c_1})(d_1+k_1+\overline{c_1})(\overline{d_1}+k_1+c_1)(d_1+\overline{k_1}+c_1)\Big] \wedge \Big[(\bar{a}+d_2)(\bar{b}+d_2)(a+b+\overline{d_2}) \,.(\overline{d_2}+\overline{k_2}+\overline{c_2})(d_2+k_2+\overline{c_2})(\overline{d_2}+k_2+c_2)(d_2+\overline{k_2}+c_2)\Big] \quad (4)$$

Using equation 4 in the first iteration, step 4 of the algorithm returns $\{a, b, k_1, k_2, c_1, c_2\}$=000101, thus the first DIP is $\vec{X_1^d}$=$\{a, b\}$=00 and corresponding oracle output is $\vec{Y_1^d}$=$\overline{0.0}$=1. Substituting $\vec{X_1^d}$ and $\vec{Y_1^d}$ in step 6 yields:

$F_2 = F_1 \wedge (\overline{d_1}.k_1) \wedge (\overline{d_2}.k_2)$ (5).

Using equation 5 in the second iteration, the while loop in step 3 of the algorithm fails, because $F_2 \wedge (c_1 \neq c_2)$ is unsatisfiable. Thus, the loop terminates and running SAT-solver on F_2 (step 9) gives $\vec{K_c}$=1, which makes the locked circuit's Boolean function $(a+b)\oplus\vec{K_c}=(a+b)\oplus 1=\overline{a+b} \neq \overline{a.b}$. Thus, $\vec{K_c}$ is *functionally incorrect*. This is an example of RE-error that improves the security.

Case 3: Figure 1 (e) shows $and \rightarrow xor$ type of RE-error, with the faulty gate shown in blue. In this case, the SAT-attack returns *UNSAT*. The while loop in logic decryption algorithm terminates after first iteration because there is no key that satisfies both the original circuit as well as the DIP. This is another example of an RE-error that improves the security, because the attacker is unable to decipher the *correct key*.

D. The Causes of the Different Cases

The causes for different cases (correct-key, incorrect-key, UNSAT) for the attack with RE-errors is three-fold:

- The DIPs ($\vec{X_i^d}$) are generated for the locked circuit, which is erroneous;
- The Outputs ($\vec{Y_i}$) are evaluated on the oracle, which provides correct outputs; and
- The locked (erroneous) circuit is constrained to satisfy the $\{\vec{X_i^d}, \vec{Y_i^d}\}$ pairs.

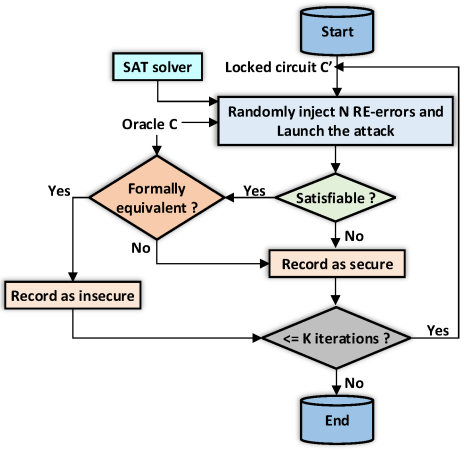

Fig. 2. Flow-chart for attack simulation with multiple-errors

Because of these contradictory constraints, unlike the no-error case, in the erroneous case the algorithm is **not guaranteed** to return *functionally correct key*. As a result, in some cases it returns *UNSAT* and in some cases *functionally incorrect key*. This motivates us to understand the likelihood of the attack success, in terms of the number of cases in which the attack fails than otherwise, in single as well as multiple error scenarios. We therefore perform extensive experiments to evaluate the attack accuracy for wide range of error scenarios.

IV. Evaluating the Impact of RE-Errors

Figure 2 shows the flow used to simulate SAT-attack. The only difference between conventional flow and this flow is the replacement of perfectly reverse-engineered circuit with erroneous reverse-engineered circuit. If the circuit has L gates, then by definition we are allowed to inject at most L errors. We inject $N \leq L$ *random RE-errors (multiple-error scenario)*, run the SAT-attack and subsequently perform formal-equivalence-checking if the result is satisfiable). The SAT-attack is recorded as *failed* for the cases when the result is *UNSAT* or when satisfiable but the decrypted key makes the locked circuit formally different from the original circuit. Otherwise it is recorded as *successful*. We run this procedure for K random multiple-error scenarios and record the statistics.

A. Evaluation Strategy for Single and Multiple Errors

To systematically understand the impact of errors on the attack accuracy, we first begin with single reverse-engineering errors. We exhaustively try all gates as candidate choices, since the total simulation time is linear in the circuit size.

Coming to multiple-errors, it is not possible to exhaust all possibilities because given an L-gate circuit, total number of possible multiple-error scenarios (combinations) is $\binom{L}{2} + \binom{L}{3} \ldots \binom{L}{L} = 2^L - L - 1$. Since this function grows exponentially with L, it is not practically feasible to exhaustively evaluate all possible multiple-error scenarios for large

TABLE II
EVALUATION OF ATTACK-SUCCESS FOR VARIOUS SINGLE REVERSE-ENGINEERING ERRORS ($N = 1$)WITH 5% LOGIC-LOCKING.

Benchmark	$nand \rightarrow nor$	$xor \rightarrow nor$	$xor \rightarrow xnor$	$xor \rightarrow nand$	$xnor \rightarrow nand$	$xnor \rightarrow nor$	$and \rightarrow or$
apex2	0%	71%	100%	71%	59%	13%	0.3%
apex4	0%	47%	100%	47%	53%	53%	0.1%
i4	0%	50%	100%	50%	57%	57%	0%
i7	0%	56%	100%	54%	61%	61%	0%
i8	0%	38%	100%	38%	56%	56%	0.1%
i9	0%	56%	100%	56%	36%	36%	0%
seq	0%	53%	100%	53%	52%	54%	0.1%
k2	0%	37%	100%	37%	57%	57%	0%
ex1010	0%	56%	100%	56%	58%	58%	0%
dalu	0%	47%	100%	48%	62%	64%	3.3%
des	0%	50%	100%	51%	56%	56%	0.1%
c432	0%	9%	22%	87%	67%	67%	0%
c499	0%	2%	38%	2%	29%	29%	0%
c880	0%	63%	100%	53%	64%	64%	0%
c1355	0%	46%	100%	46%	38%	38%	0%
c1908	0%	53%	100%	53%	68%	68%	0%
c2670	2%	48%	100%	48%	54%	54%	2.1%
c3540	7%	53%	100%	49%	50%	50%	0%
c5315	0%	35%	100%	35%	48%	48%	0%
c7552	1.1%	54%	100%	58%	53%	91%	0.3%
RISC-V	0.2%	27%	59%	27%	35%	35%	0.1%

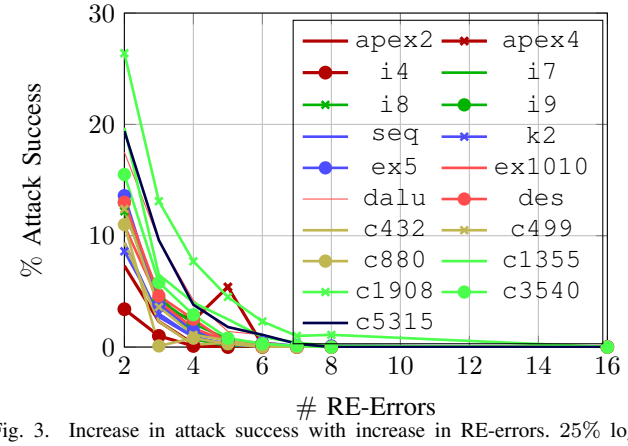

Fig. 3. Increase in attack success with increase in RE-errors. 25% logic-locking and $K = 1000$ are used for the evaluation.

industrial-strength circuits. Thus, we evaluate attack accuracy on a subset of random multiple-error scenarios.

B. Evaluation Results

IBM BladeCenter® High-Performance Cluster (HPC) dual-core nodes with $8GB$ memory, single-threaded execution and abort limit of 1 week, are used for all the runs. For each value of N ($\#RE - errors$), the experiments took 1 week. We have tried for 8 values of N $(2, 3, 4, 5, 6, 7, 8, 16)$, so it took altogether 8 weeks of compute time on the *HPC*. Table II shows the exhaustive results for single RE-errors for 8 different types of RE-errors based on the error-model described in Section III-A. We define % Attack Success as $\frac{\#\ insecure\ cases}{\#\ secure\ cases + \#\ insecure\ cases}$.

This table shows that the attack success is a function of the type of RE-error. For e.g: for $xor \rightarrow xnor$ type RE-error, attack success is 100% for most benchmarks, because the only manifestations in those cases were xor-type key gates, which the SAT-solver tolerates by flipping the corresponding key-bit (which is not possible for the remaining 7 types of RE-errors).

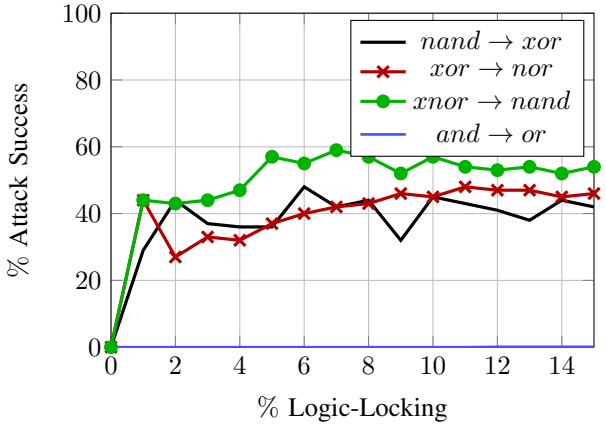

Fig. 4. Increase in %Attack Success with %Logic-Locking for k2 circuit.

On the contrary, for $nand \rightarrow nor$ and $and \rightarrow or$ types of RE-errors, attack success was close to 0% for most benchmarks. For remaining types of RE-errors, the attack success was somewhere in between these two extremes.

Figure 3 illustrates the attack success for the different values of N (X-axis) with 25% logic-locking and $K = 1000$. This plot shows exponential degradation in attack success with linear increase in N. The 25% logic-locking of RISC-V CPU using *sle* software [33] itself exceeds the abort limit, hence not reported. Figure 4 shows the attack success as a function of %logic locking. Although there is no clear trend, there is a general increase in attack success with increase in key-size. This is counter-intuitive, because we expect the attack success to degrade with increase in the key-size.

C. Evaluating the Error-Removal Attack

Attack failure is registered if the SAT-instance if unsatisfiable or if satisfiable but the decrypted key verified incorrect through functional execution. In either case, the attacker's next step would be to remove RE-errors and reconstruct the original circuit. However, the attacker is unaware of the number/locations/error-scenario, so he is forced to perform brute-force checking. If there are m possible error choices for each gate and k RE errors, the number of possibile locations is $\binom{L}{k}.(m^k)$. Coming to the number of errors, k can range from 1 to L. So, total number of possibilities is $\sum_{k=1}^{L}\binom{L}{k}.(m^k)$. We know that $\sum_{k=1}^{L}\binom{L}{k} = \mathcal{O}(2^L)$, hence the complexity of $\sum_{k=1}^{L} m^k.\binom{L}{k}$ is either similar or better than this. Thus error-removal and reconstruction is infeasible for large circuits.

V. CONCLUSIONS

In the logic-locking threat model, the adversary may not be the foundry itself but an end-user purchasing the IC from the open market. This paper analyzes such an end-user adversary, who needs to deal with errors when doing RE of the IC. We quantify the efficacy of SAT-based attack in the presence of RE-errors and identify the underlying reasons as to why it can/cannot succeed. Empirical results suggest dependence of the attack success on the type of RE-error and exponential degradation of attack success with error count. Surprisingly, in most cases the attack success increases with increase in key-size.

REFERENCES

[1] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *IEEE Design, Automation and Test in Europe (DATE)*, 2008, pp. 1069–1074.

[2] T. T. W. W. China and T. P. W. I. P. Rights. [Online]. Available: https://www.forbes.com/sites/davidvolodzko/2018/11/11/the-trade-war-with-china-and-the-problem-with-intellectual-property-rights/#9f2f079728e5

[3] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *IEEE Design Automation Conference (DAC)*, 2012, pp. 83–89.

[4] S. Dupuis, P. Ba, G. Di-Natale, M. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *IEEE International On-Line Testing Symposium*, 2014, pp. 49–54.

[5] J. Rajendran, H. Zhang, C. Zhang, G. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 21, no. 5, pp. 410–424, 2015.

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

[7] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 38, no. 2, pp. 199–207, Feb 2019.

[8] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating SAT-unresolvable circuits," in *IEEE Great Lakes Symposium on VLSI (GLSVLSI)*, 2017, pp. 173–178.

[9] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "TTLock: Tenacious and traceless logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust*, May 2017, pp. 166–166.

[10] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *ACM Conference on Computer and Communications Security*, 2017, pp. 1601–1618.

[11] U. Guin, Q. Shi, D. Forte, and M. M. Tehranipoor, "FORTIS: A comprehensive solution for establishing forward trust for protecting IPs and ICs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 4, pp. 63:1–63:20, 2016.

[12] U. Guin, Ziqi Zhou, and A. Singh, "A novel design-for-security (DFS) architecture to prevent unauthorized IC overproduction," in *IEEE VLSI Test Symposium (VTS)*, 2017, pp. 1–6.

[13] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 26, no. 5, pp. 818–830, 2018.

[14] N. Limaye, A. Sengupta, M. Nabeel, and O. Sinanoglu, "Is robust design-for-security robust enough? attack on locked circuits with restricted scan chain access," in *IEEE International Conference on Computer Aided Design*, 2019.

[15] R. Karmakar, S. Chattopadhyay, and R. Kapur, "A scan obfuscation guided design-for-security approach for sequential circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2019.

[16] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, "On the approximation resiliency of logic locking and ic camouflaging schemes," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 14, no. 2, pp. 347–359, 2019.

[17] A. Rezaei, Y. Li, Y. Shen, S. Kong, and H. Zhou, "CycSAT-unresolvable cyclic logic encryption using unreachable states," in *IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2019, pp. 358–363.

[18] S. Potluri, A. Aysu, and A. Kumar, "SeqL: Secure Scan-Locking for IP Protection," in *IEEE International Symposium on Quality Electronic Design (ISQED)*, 2020.

[19] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[20] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 95–100.

[21] T. Meade, Z. Zhao, S. Zhang, D. Pan, and Y. Jin, "Revisit sequential logic obfuscation: Attacks and defenses," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[22] K. Juretus and I. Savidis, "Time domain sequential locking for increased security," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[23] R. Karmakar, H. Kumar, and S. Chattopadhyay, "On finding suitable key-gate locations in logic encryption," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[24] R. Karmakar, S. Chattopadhyay, and M. Chakraborty, "Improving security of logic encryption in presence of design-for-testability infrastructure," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[25] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks," in *IACR Transactions on Cryptographic Hardware and Embedded Systems (CHES)*, 2019.

[26] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *IEEE Great Lakes Symposium on VLSI (GLSVLSI)*, 2017, pp. 179–184.

[27] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "ScanSAT: Unlocking obfuscated scan chains," in *IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, 2019, pp. 352–357.

[28] Y. Shen, Y. Li, A. Rezaei, S. Kong, D. Dlott, and H. Zhou, "BeSAT: Behavioral sat-based attack on cyclic logic encryption," in *IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2019, pp. 657–662.

[29] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-based attack on cyclic logic encryptions," in *IEEE International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 49–56.

[30] R. Torrance and D. James, "The State-of-the-Art in IC reverse engineering," in *IACR Transactions on Cryptographic Hardware and Embedded Systems (CHES)*, 2009.

[31] C. Bao, D. Forte, and A. Srivastava, "On application of one-class SVM to reverse engineering-based hardware trojan detection," in *International Symposium on Quality Electronic Design*, 2014, pp. 47–54.

[32] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A survey on chip to system reverse engineering," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 1, pp. 6:1–6:34, 2016.

[33] Satisfiability-based attack source code. [Online]. Available: https://bitbucket.org/spramod/host15-logic-encryption/src/default/