GateLock: Input Dependent Key-based Locked Gates for SAT Resistant Logic Locking

Vijaypal Singh Rathor, *Member, IEEE*, Munesh Singh, *Senior Member, IEEE*, Kshira Sagar Sahoo, *Senior Member, IEEE* and Saraju P. Mohanty, *Senior Member, IEEE*

***Abstract*—Logic locking has become a robust method for reducing the risk of Intellectual Property (IP) piracy, overbuilding, and hardware Trojan threats throughout the lifespan of Integrated Circuits (ICs). Nevertheless, the majority of reported logic locking approaches are susceptible to SAT-based attacks. The existing SAT-resistant logic locking methods provide a trade-off between security and effectiveness and require significant design overhead. In this paper, a novel gate replacement-based input-dependent key-based logic locking (IDKLL) technique is proposed. We first introduce the concept of IDKLL, and how the IDKLL can mitigate the SAT attacks completely. Unlike conventional logic locking, the IDKLL approach uses multiple key sequences (instead of a single key sequence) as the correct key to lock/unlock the design functionality for all inputs. Based on this IDKLL concept, we developed several locked gates. Further, we propose a lightweight gate replacement-based IDKLL called GateLock that locks the design by replacing exciting gates with their respective IDKLL-based locked gates. The security analysis of the proposed method shows that it prevents the SAT attack completely and forces the attacker to apply a significantly large number of brute-force attempts to decipher the key. The experimental evaluation on ISCAS and ITC benchmarks shows that the proposed GateLock method completely prevents the SAT-based attacks and requires an average of 56.7%, 72.7%, and 87.8% reduced area, power and delay compared to Cascaded locking (CAS-Lock) and Strong Anti-SAT (SAS) approaches.**

***Index Terms*—Intellectual Property (IP), IP Protection, Hardware Trojan, IP Piracy, Logic Locking, SAT-Attack, Anti-SAT.**

I. INTRODUCTION

DUE to globalization, intellectual property (IP) piracy, overbuilding, reverse engineering, hardware Trojan insertion, etc., are the major threats which arise during the integrated circuit (IC) life cycle [1], [2], [3]. The IC industries lose from $384 to $856 billion each year as a result of supply chain assaults [4], [5]. Nevertheless, various design techniques have been reported in the literature to address these threats, including the removal of rare-triggered nets [6], [7], logic locking [8] [9], [10], [11], and logic camouflaging [12], [13]. Over the last decade, logic locking has become the prominent approach to counter these attacks. It locks the design functionality by inserting the additional locked circuitry in the design, [14], [15] or replace the existing logic with the corresponding locked circuity [16], [17], [18], [7], [19]. The design operates correctly only when the secret key sequence is applied. This key sequence is kept in on-chip tamper-evident memory, which is beyond the reach of the attacker.

The security of logic locking relies on the key's confidentiality. Within the past five years, boolean satisfiability (SAT) [20], [21] has become the most effective attack that compromises the security of most logic locking techniques, even when utilizing a large key size, within a few minutes. This attack progressively eliminates incorrect keys by applying distinguishing input patterns (DIPs) and identifying the correct key [20], [21]. To counter this attack, various SAT-resistant logic locking methods have been proposed in the literature which integrate SAT-resistant circuits such as AES [10], SARLock [22], and Anti-SAT block (ASB) [23], [24] in the locked design. However, these methods need substantial overhead and are also susceptible to other SAT attack variants, such as removal [25], [26], App-SAT [27], and Bypass [28].

However, the Tenacious and Traceless (TTLock) [29], Stripped Functionality (SFLL) [30], cascaded locking (CAS-Lock) [31] and Strong Anti-SAT (SAS) [32] based logic locking methods have been reported to mitigate the SAT and its variants attacks. The TTLock and SFLL are vulnerable to Functional Analysis (FALL) [33], whereas the CAS-Lock is vulnerable to Identify flip s ignal (IFS) a ttack [34]. Further, the TTLock and SFLL provide a trade-off between security and effectiveness, whereas the CAS-lock provides a trade-off between removal and SAT attacks. Though SAS provides high security over other SAT-resistant methods, it may be vulnerable to IFS and removal attacks. This paper proposes a novel gate replacement-based input-dependent key-based logic locking (IDKLL) method called GateLock that effectively mitigates the SAT attacks with low overhead.

The rest of the paper is organized as follows: Section II provides the contributions of this paper. Section III presents a review of logic locking techniques. Section IV introduces a concept of IDKLL, the proposed GateLock method and its quantitative security analysis. The experimental evaluation and comparative analysis of the proposed logic locking technique are presented in Section V. Section VI concludes the paper.

II. CONTRIBUTIONS OF THE CURRENT PAPER

This section outlines the problem addressed, the proposed solution, and the novelty and significance of the solution.

V. S. Rathor is with the Department of CSE, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India, e-mail: vrathor@iiitdmj.ac.in.

M. Singh is with the Department of CSE, PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India, e-mail: munesh.singh@iiitdmj.ac.in.

K.S. Sahoo is with Department of Computing Science, Umeå University, Umeå, Sweden e-mail: ksahoo@cs.umu.se

S. P. Mahanty is with Department of Computer Science and Engineering, University of North Texas, Denton, TX, 76207, USA, e-mail: saraju.mohanty@unt.edu

A. Problem Addressed

The SAT and its variants attacks have been proven to be the most effective as they break most of the logic locking methods in a very limited time. The existing SAT-resistant techniques are either vulnerable to SAT or its variants (e.g., Removal, App-SAT, Bypass, FALL, IFS) and require significant overhead. Moreover, to the best of our knowledge, none of the existing SAT-resistant methods renders SAT-based attacks impossible; instead, they focus on increasing the attack time. This paper focuses on rendering SAT attack impossible with low overhead and protecting IC effectively during its life cycle.

B. Proposed Solution

This paper proposes a novel lightweight gate replacement-based IDKLL method called GateLock to prevent SAT-based attacks. The proposed GateLock method uses multiple key sequences (instead of a single) as a valid key to lock/unlock the design. The IDKLL divides the input patterns into multiple sets and uses a different key sequence as a correct key for each input set. No single key sequence exists in the whole key space that can provide correct output for all input sets. Based on this concept, we develop different basic IDKLL-based locked gates to lock a design by replacing its original gates. The locked design provides correct output for all sets of inputs only when their corresponding valid key sequence is applied, making it highly resistant to SAT-based attacks.

C. Novelty and Significance of the Proposed Solution

The proposed GateLock is a novel method that uses multiple key sequences to lock/unlock the design, unlike existing methods that use a single key sequence. The design locked using the proposed method provides correct output for all inputs only when their corresponding correct key sequence is applied. Unlike previous methods, the proposed method uses look-up table (LUT) based tamper-evident memory to store and retrieve the respective correct key sequence. The proposed approach makes it impossible for the SAT attack to identify the correct key because no single key sequence provides the correct functionality for all input patterns. The GateLock method tremendously increases the security against SAT and brute-force attacks with reduced overhead compared to existing SAT-resistant methods. The security analysis and experimental evaluation using ISCAS and ITC benchmarks support the effectiveness of our method.

III. SAT-RESISTANT LOGIC LOCKING: A REVIEW

Several SAT-resistant logic locking techniques such as SARLock [22] and Anti-SAT block (ASB) [23] are proposed to increase the SAT iterations or DIPs while extracting the correct key. However, they are vulnerable to removal [25], [26], App-SAT [27], double DIP [35], Bypass [28], SAT-based Signature (SigAttack) [36], and Bit-flipping [37] attacks. These methods also provide a trade-off between security (SAT attack time) and effectiveness (output corruptibility), even requiring significant overhead. Although, obfuscation of ASB using LUT-based design withholding and wire entanglement approach [18] increases its security against removal attack at the cost of large overhead [24]. A lightweight ASB design and obfuscation technique is also reported in [38] that constructs the ASB using the existing circuitry to reduce the overhead.

Further, the modified version of SARLock called TTlock [29] attempts to thwart removal attack, but it is vulnerable to App-SAT [27] and Bypass [28]. Therefore, an extension of TTLock called SFLL [30] is proposed that protects the design for multiple input patterns. The SFLL increases the output corruption and provides high security against App-SAT, double DIP and Bypass attacks. However, SFLL provides a trade-off between security and effectiveness, requires large overhead [31], [32] and also proven vulnerable to FALL [33], and structural analysis (SFLL-Unlocked) [39] attacks. To overcome the limitations of SFLL, CAS-Lock [31] and SAS [31] methods have been proposed that provide high security and effectiveness with reduced overhead. Since CAS-Lock is vulnerable to removal attacks, an extension of CAS called mirror CAS (M-CAS) is also proposed in [31]. Though M-CAS increase the security over CAS, it requires high overhead and is susceptible to Identify flip signal SAT (IFS-SAT) or Key-bit mapping SAT (KBM-SAT) attack [34].

On the other hand, the SAS block is designed using the merits of Anti-SAT and SFLL and provides high security and effectiveness against SAT attacks over SFLL. SAS requires significant overhead over the CAS-Lock, Anti-SAT and SAR-Lock and may be vulnerable to structural analysis attacks such as removal or IFS/KBM SAT as SAS uses the properties of the Anti-SAT. The comparison of the efficacy of different SAT-resistant methods against SAT variants is presented in Table I. Besides these techniques, a few other methods, such as CORruption adaptable logic locking (CORALL) [40] and DisORC [41] are also reported that provide increased output corruptibility over the SAS [32] and SFLL [30] methods. Further, generalized Anti-SAT (G-Anti-SAT) [42] and SKG-Lock+ [43] methods are reported that also provide increased output corruptibility while thwarting SAT-based attacks. However, G-Anti-SAT is vulnerable to a vulnerability assessment tool called Valkyrie [44], SigAttack [36], and a generalized attack reported in [45]. Further, all these methods focus only on increasing the SAT iterations and require increased design overhead. To the best of our knowledge, none of the existing methods focuses on making SAT attack impossible. This paper proposes a novel logic locking method (GateLock) that effectively prevents SAT attacks with low overhead.

Table I
ATTACK RESILIENCY OF DIFFERENT METHODS AGAINST DIFFERENT ATTACKS. "Y", "N" AND "Y/N" REPRESENT A METHOD MITIGATES, DOES NOT MITIGATE AND PARTIALLY MITIGATES THE SPECIFIED ATTACK.

Methods	Attacks							
	SAT	App-SAT	Bypass	double DIP	Removal	FALL	SFLL-Unlocked	IFS/KBM
SARLock	**Y**	N	N	N	N	**Y**	**Y**	**Y**
Anti-SAT	**Y**	N	N	N	N	**Y**	**Y**	N
TTL	**Y**	N	N	N	**Y**	N	N	**Y**
SFLL	**Y**	**Y**	**Y**	**Y**	**Y**	N	N	**Y**
CAS	**Y**	**Y**	**Y**	**Y**	N	**Y**	**Y**	N
M-CAS	Y/N	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	N
SAS	**Y**	**Y**	**Y**	**Y**	Y/N	**Y**	**Y**	Y/N
GateLock	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**

IV. GATELOCK: PROPOSED LIGHTWEIGHT GATE REPLACEMENT-BASED IDKLL

This section first introduces the idea of IDKLL, followed by the proposed GateLock method and its security analysis.

A. Concept of Input Dependent Key-based Logic Locking

The previous logic locking methods use only a single key sequence as a valid key to lock/unlock design (achieve correct output) for all input patterns. Thus, SAT attack identifies the correct key sequence by iteratively eliminating incorrect key sequences. *The idea behind rendering SAT attack can be that "if we develop a method where no single key is individually sufficient or correct to provide the correct output for all inputs, then, in this case, SAT attack will either eliminate all the keys or return a single key that cannot sufficient to achieve correct output for all inputs"*. Therefore, this paper introduces the idea of IDKLL, where multiple (instead of single) key sequences are used as a valid key to lock/unlock a design. We lock the circuit such that it must produce correct output for all inputs only when their corresponding correct key sequences are applied. There will not be any single key sequence that can provide correct output for all inputs, or all the key sequences are individually incorrect. Essentially, a distinct key sequence is used to unlock the design functionality for every set of input patterns. Since the correctness of a key sequence relies primarily on the inputs, it is called input-dependent key-based logic locking (IDKLL). It means a valid key sequence can only unlock the design for a particular set of input patterns. A different key sequence must be applied to unlock the functionality for a second set of inputs. The same holds for the third, fourth, and other input sets, which require other different key sequences for unlocking the functionality.

Let's consider locking a circuit that consists of n primary inputs with k key inputs using IDKLL. Assume that we utilize m key sequences out of a total of 2^k possible sequences as a valid key to lock/unlock the design. In this scenario, the 2^n input patterns need to be divided into m sets, $i.e.$, XS_1, XS_2, ..., XS_m, and the key sequences KS_1, KS_2, ..., KS_m are selected respectively as valid key for these sets. The locked design produces the correct output for all sets (XS_1, XS_2, ..., XS_m) only when the corresponding valid key sequence (KS) (KS_1, KS_2, ..., KS_m) is applied. Any other key sequence would result in an incorrect output. Importantly, there should not exist any single key sequence that can unlock the design or produce the correct output for all input patterns.

1) **Locking Example Circuit Using IDKLL:** Let us consider an example circuit as shown in Figure 1(a) to demonstrate the concept of IDKLL. First, we divide the input patterns into $m = 2$ sets, $i.e.$, $XS_1 = \{$"00", "01"$\}$ and $XS_2 = \{$"10", "11"$\}$. After that, make the original output of the circuit, $i.e.$, Y dependent on two key inputs K_1 and K_2 according to the truth table as shown in Figure 1(b). In this case, we utilize two ($m = 2$) key sequences, $i.e.$, KS_1 = "00" and KS_2 = "11", as the correct keys for the input sets XS_1 and XS_2, respectively. Finally, the locked circuit/Boolean function (Y_L), as shown in Figure 1(c), can be obtained from the truth table using Karnaugh Map (K-Map). The locked circuit provides the correct output for the input sets $XS_1 = \{$"00", "01"$\}$ and $XS_2 = \{$"10", "11"$\}$ only when KS_1 = "00" and KS_2 = "11" are applied, respectively. In addition, it must be ensured that KS_1 and KS_2 provide incorrect output for the other set of inputs to prevent a key sequence from producing the correct output for all inputs. Therefore, we set Y_L as the complement of Y ($i.e.$, $Y_L = !Y$) for the patterns {"0010", "0011", "1100", "1101"}.

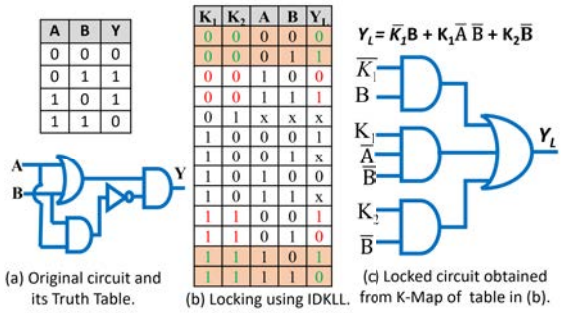


Figure 1. Locking a circuit using IDKLL. Locked circuit provides correct output for the input sets {"00", "01"} and {"10", '11"} only when the key sequence K_1K_2 = "00" and "11" are applied respectively as a correct key.

It is analyzed that locking a design requires low implementation cost if we set output values don't care in case of incorrect (remaining) key sequences, $i.e.$, "01" and "10". However, in this scenario, the expression of Y_L obtained from the K-map does not rely on all the inputs (A and B) due to optimization. Therefore, we replace the "don't care" (x) values in Y_L with '1'/'0' for specific patterns, such as "1000" and "1010". Nonetheless, the designer can also choose alternative patterns to achieve this. If the designer is not concerned about the dependency, then "don't care" values can be maintained for the remaining key sequences. If the locked functionality already exhibits dependency on all input patterns while using "don't care" values for the remaining key sequences, those values can be retained to obtain an optimized locked design. Further, it is also observed that using incorrect output (!Y) instead of "don't care" (x) for the remaining key sequences results in higher output corruption at the cost of increased overhead. Thus, the designer can achieve the desired level of output corruption in this technique, which may not be possible with existing logic locking techniques.

2) **LUT-based Tamper-Proof Memory for IDKLL:** Since the proposed method uses multiple key sequences as a valid key for unlocking the design, we employ LUT-based tamper-evident memory (instead of conventional tamper-proof memory) that can store all selected KSs and provide respective correct KS for an input set. This LUT-based memory allows fetching a correct KS corresponding to a specific set of input patterns. The implementation of this memory is based on the LUT-based memory as used in [17], [18]. However, using LUT-based tamper-proof memory incurs more overhead compared to the conventional normal tamper-proof memory because it stores multiple key sequences that are valid for different input sets. For storing m key sequences, the LUT-based memory requires $m\times$ storage space compared to conventional tamper-proof memory. In conventional memory, only a single

key sequence is stored that is valid for all inputs. In addition, LUT-based memory uses an additional multiplexer unit to extract the respective valid key for different input sets, which also increases the overhead. Further, using LUT-based tamper-proof memory may also introduce vulnerability or security issues. One can also use the wire entanglement approach to enhance security [18], [24]. However, overhead and security analysis of LUT-based temper-proof memory is out of scope in this paper. Here, it is assumed that the LUT-based tamper-proof memory is implemented in a trusted facility, inaccessible to the attacker and the designer activates it after fabrication [3], [9], [14], [15]. We use LUT-based memory as one solution to store multiple key sequences and retrieve respective key sequences on applying an input. One can also explore other cost-effective solutions to do the same.

Figure 2(a) illustrates the implementation and integration of this LUT-based tamper-proof memory with the IDKLL-based locked circuit (Figure 1).

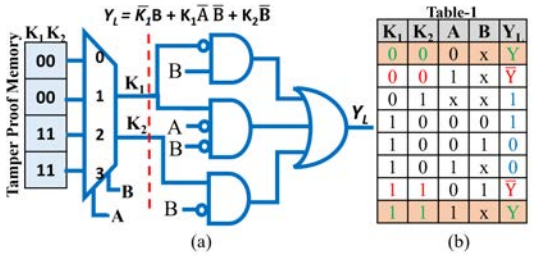


Figure 2. LUT-based tamper-proof memory and its integration with IDKLL-based locked circuit. (b) The deterministic functionality of locked circuit.

It can be observed from Figure 2(a) that the correct key sequence "00" can be readily extracted from this memory by applying inputs {"00", "01"} to the selection lines of the multiplexer. The diagram shows that the key sequences "00" and "11" are stored multiple times in memory, potentially increasing the design cost. Thus, an alternative approach is to store only unique valid key sequences, reducing the memory requirements. However, the designers can also intentionally introduce unused inputs or other internal wires to the tamper-proof memory using the wire entanglement technique [18] to obfuscate memory inputs and enhance security. Including additional elements makes it hard and creates a dilemma for potential attackers while distinguishing between genuine inputs and those added for obfuscation purposes.

Furthermore, it can be observed from the locked functionality mentioned in truth tables displayed in Figure 1(b) that we retain some values as don't care in the outputs (Y_L). However, the obtained locked circuit/Boolean function always produces deterministic output for each key-input pattern. Therefore, we present Table-1 in Figure 2(b), which shows a deterministic output for each input pattern of the locked circuit/Boolean function as depicted in Figure 1(b). The analysis of Table-1 shows that the IDKLL-based locked circuit produces a constant output of '1' for the key sequence "01". While this output may be correct for a few inputs, it is not accurate for all input patterns. Hence, the designer may also utilize the KS "01" along with other KS to obtain the correct output for all inputs. However, in this case, no single key sequence can provide the correct output for all inputs, rendering SAT attacks impractical for the attacker. This is because our approach ensures that applying a DIP generates different outputs with two key sequences even belonging to the valid set. This causes the SAT solver to eliminate one correct key sequence if it produces an incorrect output for that DIP. The same elimination process occurs with subsequent DIPs. Ultimately, the SAT solver either eliminates all the key sequences without finding the correct key or ends up with a single key sequence that is insufficient to produce the correct output for all inputs.

3) **SAT Attack Prevention using IDKLL:** Table II demonstrates the example of SAT attack on the IDKLL-based locked circuit as shown in Figure 2. It can be observed that applying a DIP "01" provides correct output for the KS "00" and "01". Whereas it provides incorrect output for KS "10" and "11". Thus, this DIP eliminates the KSs "10" and "11". Similarly, in the second SAT iteration, the KS "00" provides wrong output for the DIP "10", thus it is eliminated. Similarly, the KS "01" is also eliminated with DIP "11" in third iteration. In this way, the SAT attack eliminates all the KSs, and none of the KS remains correct. Hence, the attacker cannot identify the correct key, which makes it impossible for the attacker to render an SAT attack on the proposed IDKLL.

Table II
SAT ATTACK ON IDKLL-BASED LOCKED CIRCUIT SHOWN IN FIGURE 2

Input A	Input B	**Y**	Y with different KS **00**	Y with different KS 01	Y with different KS 10	Y with different KS **11**	Elimination of Key Sequences
0	0	**0**	**0**	1	1	1	
0	1	**1**	**1**	**1**	0	0	itr-1: {"10", "11"}
1	0	**1**	0	**1**	0	**1**	itr-2: {"00"}
1	1	**0**	1	1	**0**	**0**	itr-3: {"01"}

The implementation of the proposed IDKLL for a small circuit is presented above. However, locking a practical size entire design exhibiting multiple inputs and outputs, employing the aforementioned procedure becomes exceedingly intricate and arduous. Additionally, it may require substantial overhead, rendering it impractical in certain cases. Therefore, we introduce a cost-effective gate replacement-based IDKLL called GateLock for locking a design in the next subsection.

B. GateLock: Gate Replacement Based IDKLL

Since a digital circuit is mainly constructed with AND, NAND, OR, NOR, XOR and XNOR, it can be locked by replacing its original gates with corresponding locked gates, which are locked using IDKLL. To achieve this, we first construct the locked circuitries of these basic gates using the proposed IDKLL. Further, we lock the design by replacing the original gates with the corresponding locked gates/circuitries.

1) **IDKLL-based Locked Gates:** The locked circuitries of 2-input AND, NAND, OR, NOR, XOR and XNOR gates are shown in Figure 3. These locked gates are obtained by solving the K-Map after locking their original functionality using two key inputs. The Truth Table for obtaining the locked circuitries (Figure 3) of AND, NAND, OR, NOR, XOR and XNOR gates using K-Map is shown in Table III. Here, Y_1, Y_2, Y_3, Y_4, Y_5,

and Y_6 represent the outputs considered while locking AND, NAND, OR, NOR, XOR and XNOR gates, respectively.

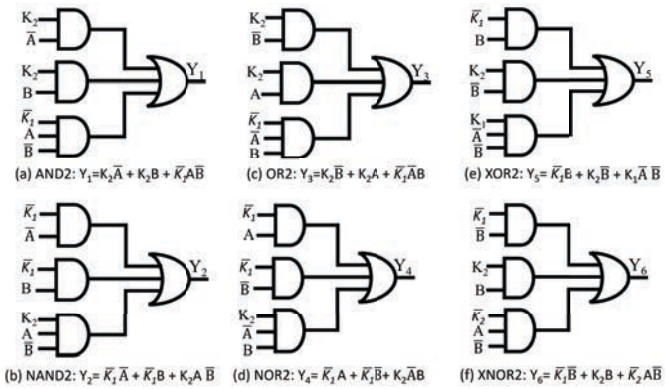


Figure 3. The locked circuitries of 2-input basic gates with two key inputs. These locked gates are obtained by locking the functionality of gates using the proposed IDKLL. The correct key sequences are K_1K_2 = "00" and "11" for the inputs sets AB = {"00", "01"} and {"10", "11"} respectively.

Table III
TRUTH TABLE FOR LOCKING BASIC GATES

K_1	K_2	**A**	**B**	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
0	0	0	0	0	1	0	1	0	1
0	0	0	1	0	1	1	0	1	0
0	0	1	0	1	0	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	x	x	x	x	x	x	x	x
1	0	0	0	x	x	x	x	1	0
1	0	x	1	x	x	x	x	x	x
1	0	1	0	x	x	x	x	0	1
1	1	0	0	1	0	1	0	1	0
1	1	0	1	1	0	0	1	0	1
1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	0	1

We lock the functionality of these gates by dividing the input combinations into two sets XS_1={"00", "01"} and XS_2={"10", "11"}. A gate provides correct output for XS_1 and XS_2 by applying the key sequences KS_1 = "00" and KS_2 = "11", respectively. Applying KS_1 and KS_2 for the other inputs provides incorrect output. We assume that 'don't care output values' for all the input combinations for the other key sequences ("01" a361nd "10") to get optimized locked circuitries. However, it is analyzed from the K-Map solver that if we keep don't care output for all inputs, the locked functions of XOR and XNOR gates do not depend on all inputs, *i.e.*, K_1, K_2, A, B. Hence, we set Y_5 and Y_6 as '1'/'0' and '0'/'1' for the key and input combinations "1000" and "1010" to ensure the dependency of output on all the inputs. The correct output for all input sets can be achieved only when their respective valid key sequences are applied.

Though we used KS_1 = "00" and KS_2 = "11" as valid key sequences for the input sets XS_1 and XS_2, respectively, we also tried different combinations of key sequences while developing the above IDKLL-based gates. During the study, it is observed we can also use other key sequences and/or other input sets to obtain such locked circuitries. For example, we can obtain the similar IDKLL-based locked AND gate functionality (i.e., $Y_1 = \overline{K_2}.\overline{A} + \overline{K_2}.B + \overline{K_1}.A.\overline{B}$) can also be developed using KS_1 = "01" and KS_2 = "10" as valid key sequences for the input set "00", "01" and "10", "11", respectively. It can be observed that both the expressions of locked AND gate exhibit the same number of literals, which shows that using different key sequences requires approximately the same cost. Since the attacker is unaware of which key sequences are used as a valid key, it will introduce non-determinism and increase search space, which will make it harder for the attacker to know the correct respective locked circuitries or key sequences. On the other hand, using uniform keys may introduce vulnerability, and it may help the attacker in finding the correct respective locked circuitries and their valid key sequences.

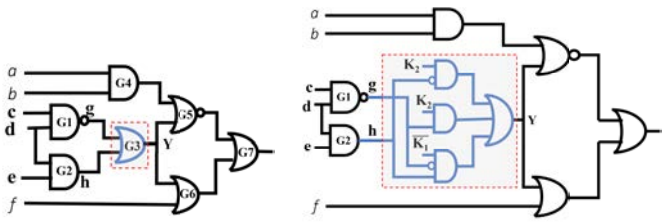
Since we use don't care (x) in the outputs for the key sequences "01" and "10", the obtained locked gates may provide correct output on applying a single key sequence for all patterns due to deterministic logic. Therefore, the designer has to verify it while designing locked gates using don't care. We have also verified and ensured that these locked circuitries do not provide correct output with a single KS for all inputs. However, this verification time can be computationally expensive if it is required to verify entire/large circuit for all input/output instances. Since the proposed method locks the entire circuit by replacing its original gates with the IDKLL-based locked gates, it is required to verify only IDKLL-based gates before replacement. Verifying all locked gates individually for all input/output instances is a one-time process, and it is not computationally expensive for a designer. The Truth Table for all the proposed locked gates (Figure 3) with deterministic output is shown in Table IV. It can be observed from this table that there is no such single KS that can provide the correct output for all inputs. The proposed locked gates only provide correct output for all inputs when their respective correct key sequence (KS_1 or KS_2) is applied.

Table IV
TRUTH TABLE OF LOCKED BASIC GATES/CIRCUITRIES

K_1	K_2	**A**	**B**	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
0	0	0	0	0	1	0	1	0	1
0	0	0	1	0	1	1	0	1	0
0	0	1	0	1	0	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	x	x	1	1	1	1	1	1
1	0	0	0	0	0	0	0	1	0
1	0	x	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1
1	1	0	0	1	0	1	0	1	0
1	1	0	1	1	0	0	1	0	1
1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	0	1

2) **Locking a Design using IDKLL-based Locked Gates:** The designer can lock the design by judiciously replacing the existing design gates with their respective IDKLL-based locked gates. In this paper, we randomly select different design gates to lock the functionality of a large design using proposed input-dependent key-based logic locking. Further, the selected gates are replaced with their corresponding proposed locked circuitries/gates. The example of locking a sample circuit by replacing an OR gate (G3) with a proposed IDKLL-based locked OR gate is shown in Figure 4.

The selection of gates for locking the design can be either random or strategic. Random selection may introduce some



(a) Originsl Circuit. (b) Locked circuit after replacing a gate G3 with its IDKLL based locked OR gate

Figure 4. Locking an example circuit using the proposed GateLock method.

level of unpredictability during the locking process, but it may not achieve the desired output corruption. On the other hand, a strategic replacement of gates can lead to a higher degree of output corruption. For instance, selecting gates with the highest fault impact, as described in [15], can greatly enhance output corruptibility. The designer can also combine both approaches to achieve a higher level of security for the locked design.

Further, the proposed method does not use a specific structure of locked gates as used in exiting approaches [16]. [15]. The proposed locked gates/circuitries look like another design circuit. Thus, it would be very difficult to correctly identify the replaced/locked circuitry in the whole design. However, it may be possible for a motivated attacker to identify an individual structure of a locked gate in the design, re-synthesizing a locked design (i.e., after replacement) as done in previous methods [24] [31], [10], [18], [38] will make it hard for the attacker to identify locked structures correctly. Further, due to interference of the output of the locked gates in the locked design, it may not be feasible for the attacker to know the output of a locked structure individually by applying input patterns [13], [46], [10]. Deciphering a locked structure in a locked design will be very hard for a motivated attacker. This paper presents the locked gates for standard logic gates. One can also generate the locked circuities for the other small logics/circuitries using IDKLL. Additionally, the designer can lock the design by replacing the small circuitries with their corresponding locked circuitries. This will make it hard for the attacker to distinguish which parts of the circuit are replaced with which locked circuits. The quantitative security analysis of our method is presented in the next subsection.

C. Security Analysis of Proposed Method

This section first presents the analysis of the proposed method against SAT and brute-force attacks followed by other attacks such as key-sensitization, removal, App-SAT, double DIP, Bypass, SAT-based Signature and Bit-flipping.

1) Threat Model or Information Available to Attacker: We consider the same threat model as considered in the previous logic locking methods [32], [34], where the attacker may reside in an untrusted foundry, or he/she may be the end user who has the capability and EDA tools to extract the gate level netlist from the fabricated chip. Therefore, the attacker can have access to (i) a locked gate-level netlist and (ii) an activated chip bought from the open market. The attacker tries to infer the circuit functionality or secret key to unlock the design using the above resources. Since the designer is the defender who activates the chip and only knows the secret key (key/input-set mapping), it is assumed that LUT-based tamper-proof memory is inaccessible to the attacker and the attacker is unaware of the number of key sequences, which are their specific values and their mapping to internal inputs of the circuit, i.e., key/input-set mapping. It means the attacker will also be unaware of the number of inputs associated with each valid key set.

2) Analysis Against SAT and Brute-force: Since the SAT attack cannot decipher the correct key from the IDKLL-based locked circuit, the attacker has to employ a brute force attack to identify the valid key. Hence, we conduct a quantitative security analysis of the proposed method against the brute force attack. If a design containing n primary inputs is locked using conventional logic locking methods with k key inputs (k-bit), the attacker would need to employ either exhaustive key sequences or brute force attempts to decipher the design functionality [15], [10], [47]. If total brute force attempts are denoted with b, then b can be expressed as

$$b = 2^k \tag{1}$$

Using SAT-resistant methods such as Anti-SAT [23], [24] compels the attacker to make a minimum of λ attempts/iterations to extract the correct key. The value of λ for a k-bit key can be represented by Equation (2) [3].

$$\lambda = 2^{k/2} \tag{2}$$

On the other side, if we select m key sequences from the total of b to lock the design using the proposed GateLock method, then the secret key will be represented as the knowledge of selected m key sequences and their mapping to input-sets (which key sequence corresponds to an input set). To know this secret key, the attacker would require two things: (i) knowledge of the corresponding output for each key sequence to get the possible correct key sequences and (ii) correct mapping of valid key sequences to input sets (i.e., correct combinations of key sequence and input). The number of attempts required to determine the output of each key sequence would be equivalent to the number of brute force attempts, $i.e.$, $b = 2^k$ as shown in Equation (1). As the attacker is unaware of the number of key sequences (value of M), which specific key sequences constitute a valid key set, and what is their mapping with the input sets? Hence, the attacker must apply all possible permutations of the b key sequences to identify the valid set of key sequences and their mapping (combinations of sequences and inputs) to input-sets. If we denote the total number of permuted combinations for the b key sequences as PC, it can be represented as follows.

$$PC = {}^bP_1 + {}^bP_2 + {}^bP_3 + ... + {}^bP_b \tag{3}$$

The Equation (3) can also be expressed as follows.

$$PC = b! \sum_{r=1}^{b-1} \frac{1}{(r)!} \tag{4}$$

Now, the total number of brute force attempts required to determine the correct key sequences and their mapping with the input sets will be equivalent to PC. However, the validity/correctness of each permuted combination (mentioned in Equation 4) is also required to know the output of each key sequence. Here, the validity denotes whether a permuted combination can provide the correct output for all patterns or not. In the worst case, all permuted combinations need to be checked. The number of iterations required for the knowing output of each key sequence is $b = 2^k$. Hence, total attempts will be calculated by adding b and PC as follows:

$$Total\_Attempts = b + PC \tag{5}$$

Substituting the values of b and PC from above, then

$$Total\_Attempts = 2^k + (2^k)! \sum_{r=1}^{2^k-1} \frac{1}{(r)!} \tag{6}$$

Now, it can be easily observed from Equation (1) and Equation (2) that our method significantly increases the number of attempts required to reveal the secret key than previous techniques. The increase in the number of brute force attempts can be calculated by subtracting Equation (1) from Equation (6), as illustrated below.

$$Increased\_Attempts = (2^k)! \sum_{r=1}^{2^k-1} \frac{1}{(r)!} \tag{7}$$

Besides the above, it is also possible that the attacker may reframe the problem solved by SAT-based algorithm by learning the multiple key sequences and their associated input sets. In this case, the algorithm would require to learn all combinations of key sequences and inputs. Further, designing SAT algorithm that learns these combinations will not be applicable to our method as the attacker is unaware of which valid key sequence is correct for which input. Hence, the attacker has to employ all permuted combinations as mentioned in Equations (4) and (6). For a practical size circuit, b will be sufficiently large; thus, this cannot be removed from the analysis. In conclusion, it is evident from the above that our method effectively prevents SAT-based attacks and substantially augments the required number of brute force attempts compared to previous logic locking methods.

3) Analysis Against Other Attacks: The proposed GateLock method not only prevents the SAT attacks but it also effectively mitigates other attacks such as key-sensitization [10], removal [25], [26], App-SAT [27], double DIP [35], Bypass [28], Bit-flipping [37] and SAT-based Signature (SigAttack) [36]. The key-sensitization attack determines the secret key by propagating the key values one by one at output. However, it can be applied only when the effect of one key-input does not interfere with another key-input. In the proposed GateLock methods, every IDKLL-based locked gate exhibits two interfering key-inputs. Due to this interference, the attacker cannot sensitize and propagate the key values at the output using a key-sensitization attack. In addition, there are different valid key sequences for different inputs; therefore, sensitizing and propagating one key value for the applied input cannot be correct for the other inputs. Hence, the sensitization cannot be applied to the proposed GateLock method.

Besides the sensitization, the removal and Bit-flipping attacks can also not be applied to the proposed method. The removal attack identifies and removes the inserted SAT-resistant block from the locked design using structural and signal probability skewed (SPS) analysis. In contrast, the Bit-flipping attacks attempt to separate the key-inputs of the inserted point function from the key-inputs of the primitive and conventional logic locking and then apply the conventional SAT attack to determine the correct key. The application of both these attacks is based on the specific structure of the inserted SAT resistance block and the conventional SAT attack. Since the proposed IDKLL-based gates are SAT resistant, they do not follow any specific structure and do not exhibit SPS values different from the other nets of the original circuits. Further, we used our gates to replace the existing original circuitry; therefore, it will approximately be impossible for the attacker to know the replaced circuitry and remove the proposed gates from the locked design. Hence, the removal and Bit-flipping attacks cannot be applied to the proposed GateLock.

The proposed GateLock can also effectively mitigate the other variants of SAT attack, such as App-SAT, double-DIP, and Bypass. All these attacks exploit the properties of i) inserted point function block and ii) low output corruptibility or error rate of the previous SAT-resistant methods. Similar to these attacks, Signature-based SAT attack (SigAttack) is also applicable in point function-based methods like Anti-SAT [24], SARLock [22] and G-Anti-SAT [42]. SigAttack determines the key-revealing Signature of the locked circuits and then uses them to decipher the key. The proposed method is not based on the point function and can easily provide desirable or high output corruptibility. Further, the replacement of original gates with IDKLL-based locked gates does not leave any signature for the attacker. Therefore, these attacks cannot be applied to the proposed method. Further, using different valid key sequences for different input sets makes the SAT attack and its above variants impossible on the proposed GateLock method.

The experimental evaluation of the proposed GateLock method is presented in the section.

V. Experimental Results and Discussion

This section presents the simulation setup, results, and a comparative analysis of our GateLock method.

A. Experimental Setup

The proposed GateLock method is implemented by embedding the varying number of keys ($i.e.$, 16, 32, 64, 128) in the ISCAS and ITC benchmark circuits. The security of our GateLock is evaluated by performing SAT attack [20] on the locked benchmarks ($i.e.$, Prop_16K, Prop_32K, Prop_64K, Porp_128K). Further area, power, delay of GateLock are extracted using Cadence RTL compiler with 45nm Nangate Open Cell Library [48] and compared with the well-known previously proposed methods, namely Anti-SAT block (ASB) [23], TTLock [29], SFLL [30], CASLock [31], and SAS [32].

B. SAT Attack Results and Discussion

The security of the proposed GateLock is evaluated against SAT attack by applying SAT attack [20] on ISCAS and ITC benchmarks locked using GAteLock. As a result, we found that the SAT attack cannot extract the correct key from any of the locked benchmark circuits. It completely failed to identify the correct key sequences even with applying the Partial-Break algorithm along with fault analysis [15]. The SAT attack provides the "UNSAT Model" as output and returns the key as "xxxxx..x" as shown in Figure 5.

```
vijay@vijay-VirtualBox:~/SAT9/bin$ ./sld ../benchmarks/Locked_Gates/
c7552_Gate16.bench ../benchmarks/myoriginal/c7552.txt
inputs=207 keys=16 outputs=108 gates=3560
iteration: 1; vars: 17790; clauses: 3130; decisions: 1066
iteration: 2; vars: 23532; clauses: 3500; decisions: 1454
        :
iteration: 8; vars: 57984; clauses: 5036; decisions: 3552
finished solver loop.
UNSAT model!
key=xxxxxxxxxxxxxxxx
iteration=8; backbones_count=0; cube_count=143888; cpu_time=0.168011
; maxrss=23.8047
```

(a) SAT attack results on c7552 locked circuit with 16 keys.

```
vijay@vijay-VirtualBox:~/SAT9/bin$ ./sld ../benchmarks/Locked_Gates/
c880_Gate16.bench ../benchmarks/myoriginal/c880.txt
inputs=60 keys=16 outputs=26 gates=431
iteration: 1; vars: 2477; clauses: 817; decisions: 214
iteration: 2; vars: 3245; clauses: 1067; decisions: 214
finished solver loop.
UNSAT model!
key=xxxxxxxxxxxxxxxx
iteration=2; backbones_count=0; cube_count=5848; cpu_time=0.022509;
maxrss=8.09375
```

(b) SAT attack results on c880 locked circuit with 16 keys.

```
vijay@vijay-VirtualBox:~/SAT9/bin$ ./sld ../benchmarks/Locked_Gates/
c7552_Gate32.bench ../benchmarks/myoriginal/c7552.txt
inputs=207 keys=32 outputs=108 gates=3608
iteration: 1; vars: 18086; clauses: 21057; decisions: 0
finished solver loop.
UNSAT model!
key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
iteration=1; backbones_count=0; cube_count=32548; cpu_time=0.064352;
 maxrss=17.4922
vijay@vijay-VirtualBox:~/SAT9/bin$
```

(c) SAT attack results on c7552 locked circuit with 32 keys.

Figure 5. Screen shots of the demonstration of SAT attack on the benchmarks locked using the proposed GateLock method.

It can be observed from this figure that the application of SAT attack on locked c880 and c7552 benchmark provides "UNSAT Model", which means it fails to determine the correct key. Besides the above, we have also tested other locked benchmarks for SAT attacks; in all the cases, SAT attack returns the "UNSAT Model" as output. The sample SAT attack results of the proposed GateLock are shown in Table V. The 4^{th} and 7^{th} columns of this table represent the output of SAT solver, whether the key is found or not (i.e., UNSAT Model). Further, it is also analyzed that the SAT attack never identifies the correct key in the proposed GateLock method, even locking with any number of keys. This is because of using multiple key sequences as a valid key.

On the other hand, due to the use of a single key sequence as a correct key in Anti-SAT block (ASB), CAS-Lock, and other SAT-resistant methods, the SAT attack runs until the worst-case time and then returns the correct key. In most existing methods, the SAT attack retrieves the correct key in an average 2^k iterations if the design is locked with $2k$ keys. However, the time and #SAT iterations in our GateLock method are significantly low due to the unsuccessful termination of SAT attack. Therefore, the attacker has to employ a brute force attack only to decipher the correct key. The comparison of brute force attempts and SAT iterations required by proposed and existing methods is shown in Table VI.

Table V
SAMPLE RESULTS OF SAT ATTACK DEMONSTRATION ON BENCHMARKS LOCKED USING PROPOSED GATELOCK METHOD

Circuits	Prop_16K			Prop_32K		
	#Itr.	Time (Sec.)	Output	#Itr.	Time (Sec.)	Output
c432	8	0.023494	UNSAT Model	5	0.19515	UNSAT Model
c499	4	0.016862		30	0.293555	
c880	2	0.022509		1	0.011458	
c1355	1	0.014087		26	0.156574	
c1908	1	0.017569		1	0.016471	
c2670	9	0.064356		7	0.058989	
c3540	3	0.063234		2	0.070557	
c5315	10	0.134589		2	0.09265	
c7552	8	0.168011		1	0.064352	

Table VI
COMPARISON OF #SAT ITERATIONS ($k = 2 * n$ KEYS, $m = 8$) AND #BRUTE FORCE ATTEMPTS IN EXISTING AND PROPOSED METHODS FOR VARYING KEY SIZE. m DENOTES THE SELECTED CRITICAL INPUTS.

n	SARLock	ASB	TTLock	CAS	SAS	Proposed GateLock
	2^k	2^n	2^k	2^n	$\frac{2^n+m}{2}$	$2^k+(2^k)!\sum_{r=1}^{2^k-1}\frac{1}{(r)!}$
1	4	2	4	2	5	**44**
2	16	4	16	4	6	**3.60E+13**
3	64	8	64	8	8	**2.18E+89**
4	256	16	256	16	12	**1.47E+507**
5	1024	32	1024	32	20	**9.31E+2639**
6	4096	64	4096	64	36	**6.26E+13019**
7	16384	128	16384	128	68	**2.07E+61936**
8	65536	256	65536	256	132	**8.87E+287193**

It can be observed from this table the number of brute force attempts is significantly high in our method compared to the SAT iterations required in existing methods. Suppose an SAT iteration and one attempt require the same time. In that case, the attacker needs a significantly long time to decipher the correct key in GateLock over all the existing techniques for the same number of keys. For $k = 6$, the SARLock and TTLock require 64 iterations, and the ASB, CAS and SAS require only 8 SAT iterations. Whereas the GateLock requires 2.18E+89 attempts, which is significantly higher than the existing methods. Further, we also analyze the security of the proposed GateLock method against other SAT variants such as App-SAT, removal and IFS. It is almost impossible to defeat the proposed GateLock method with these attacks because our method is not based on a single-point function and can provide the desired output corruptibility. The proposed GateLock method provides an effective solution to mitigate all such attacks with less overhead. The overhead analysis of the proposed GateLock method is presented in the next subsection.

C. Overhead Results

We analyze the implementation overhead of our GateLock method while embedding varying key sizes (in bits), $i.e.$, 16, 32, 64, 128 in different ISCAS and ITC benchmarks. The original and locked benchmark circuits are synthesized using the Cadence RTL compiler, and different design metrics are extracted. It is observed from the results that the proposed GateLock requires a large overhead on small benchmarks

Table VII
DESIGN METRICS OF PROPOSED GATELOCK METHOD WHILE EMBEDDING VARYING NUMBER OF KEYS IN LARGE ISCAS AND ITC BENCHMARKS

Circuit	Area (μm^2))			Power (nW)			Delay (ps)		
	Prop_32K	Prop_64K	Prop_128K	Prop_32K	Prop_64K	Prop_128K	Prop_32K	Prop_64K	Prop_128K
s35932	12365	12408	12480	842474	847504	855824	3057	3129	3421
s38417	11850	11868	11929	666947	729550	735333	5686	5790	**5643**
s38584	9917	9949	10083	627943	630676	640410	4066	**3979**	4093
b14	3490	3612	3702	259701	293536	304253	18931	20580	21921
b15	6577	6664	7070	336262	350124	373289	21982	**21359**	22931
b17	20821	20867	20878	1046983	**1032300**	1074434	20594	21336	**20914**
b18	51778	52163	52272	3024906	3058028	3073839	14972	15604	15819
b20	7307	7357	7426	489337	498038	533511	22535	**22284**	23866
b21	7568	7569	7680	520702	522752	**507265**	22518	23074	**22956**
b22	10920	10997	11064	727418	742219	795411	23071	**22869**	23348

(ISCAS-85), even locking with a small key size. Whereas it requires significantly l ow a rea, p ower a nd d elay f or locking large benchmarks, even with large key sizes. The results for locking large-size ISCAS-89 and ITC benchmarks using the GateLock method are shown in Table VII. It is clear from the last row of this table that the proposed GateLock method on average requires area, power delay as $14458\mu m^2$, $889357nW$ and $16491ps$, respectively for while locking a design using 128-bit key. It can also be observed that the average of area, power and delay values are only slightly increased while increasing key size from 32 to 64 and then 128. Even in some cases, values are also decreased as mentioned with bold. It means our GateLock method will also not require high implementation cost, even locking a design with a more significant s ize k ey, i .e., K =256 o r K =512. T he comparison of the proposed Gate-Lock method with well-known existing methods is presented in the next subsection.

D. Comparisons

We implement the existing well-known SAT-resistant logic locking methods $i.e.$, ASB [23], CAS [31], and SAS [32]. The design metrics of the proposed method are compared with these methods. The comparison of average area, power and delay required by proposed GateLock and existing ASB, CAS and SAS methods while inserting varying keys in large ISCAS-89 and ITC benchmark circuits is presented in Table VIII. It can be observed from this table that the ASB requires low average area, power and delay compared to CAS and SAS methods. Though the SAS method requires high implementation cost, it provides high security against SAT attacks over the ASB and CAS insertion-based methods. The ASB, CAS and SAS methods are implemented without any obfuscation; otherwise, the required implementation cost will be further increased in these methods. Further, these methods may be vulnerable to removal or structural analysis attacks if implemented without obfuscation [26], [34]. At the same time, our method requires very low implementation costs compared to all these methods for varying key sizes.

The comparison of the required average overhead (%) of GateLock and ASB, CAS and SAS methods is presented in Figure 6. It is clear from this figure that the overhead for CAS and SAS methods is approximately the same, but it is slightly more than the overhead of ASB. The proposed method requires significantly low overhead over all the existing methods for all the mentioned key sizes. For embedding 128 keys in the benchmarks, the SAS requires 3.5%, 3.75% and 28% area, power, and delay overhead, whereas GateLock requires only 1.43%, 1.08%, and 3.6% area, power, and delay overhead which is 59%, 71%, and 87% less, respectively.

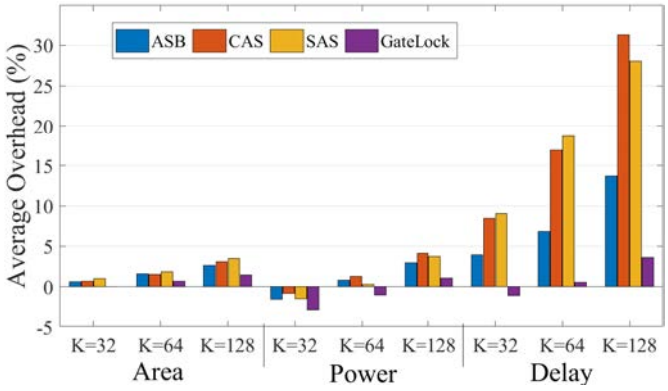


Figure 6. Comparison of average overhead (%) required by proposed and existing methods for embedding K=16, K=32 and K=128 keys in the large ISCAS-89 and ITC benchmarks.

Besides the above, we also compare the implementation cost of the proposed GateLock method with TTLock [29] while embedding 32 and 64-bit keys in large ISAS-89 benchmark circuits (s35932, s38417, s38584). On average, the TTLock requires 2.5%, 3.4%, and 0.7%, whereas the proposed GateLock requires 0.4, -0.9 and 8.1% area, power and delay overhead, respectively. However, the delay overhead in our method is higher than the TTLock because of the large delay overhead in the s35932 benchmark. For the average of s38417 and s38584 circuits, the GateLock method requires a significantly lower delay than the TTLock. It is reported that the SFLL is the extension of TTLock and provides high security over TTLock but requires more overhead over the TTLock, CAS, and SAS [30], [31], [32]. Since it is analyzed in the above discussion that GateLock requires low overhead over all these methods, our method will also require lower overhead than the SFLL method for the same size key. In addition, it is also observed in the literature that other recently developed SAT-resistant methods, i.e., CORALL [40], DisORC [41], G-Anti-SAT [42] and SKG-Lock+ [43] provide better security compared to the SFLL, CAS and SAS methods, but they also incur slightly increased overhead. Most of these methods incur around 10%

Table VIII
COMPARISON OF AVERAGE AREA, POWER AND DELAY REQUIRED BY PROPOSED GATELOCK AND EXISTING METHODS

Method	Area (μm^2)			Power (nW)			Delay (ps)		
	K=32	K=64	K=128	K=32	K=64	K=128	K=32	K=64	K=128
ASB	14339	14477	14633	866187	887057	906152	16550	17005	18098
CAS	14350	14475	14698	872427	891093	916603	17268	18617	20898
SAS	14396	14514	14755	866754	882127	912844	17360	18894	20379
GateLock	**14259**	**14345**	**14458**	**854267**	**870473**	**889357**	**15741**	**16000**	**16491**

area overhead in the average case with the same or less key size [43]. On the other hand, our method requires less than 4% area overhead with a 128-bit key size. Hence, the proposed method will also require low overhead over these methods.

The above comparison is presented for the same key size, but our method achieves $2^k\times$ higher security (Equation 7) with the same size key over the ASB, CAS and SAS methods. It means our method can provide the same security only using half key size compared to ASB, CAS and SAS. In this case, the required overhead of GateLock will be reduced further compared to existing methods. For example, the CAS/SAS methods require 3.1%/3.5%, 4.2%/3.8% and 31.3%/28.0% area, power and delay overhead to achieve 2^{64} SAT iterations, whereas, GateLock achieves the same security (2^{64}) with only 0.64%, -1.06%, and 0.52% overhead, respectively. Hence, our GateLock method provides high security against SAT-based attacks while reducing design overhead significantly.

VI. CONCLUSION

This paper proposes a novel SAT-resistant logic locking method called GateLock that effectively prevents SAT-based attacks. The GateLock method is based on the concept of IDKLL, where multiple (instead of a single) key sequences are utilized as the correct key to lock/unlock the design functionality for all inputs and mitigate the SAT attack completely. In our method, we proposed different IDKLL-based locked gates, which are used to lock the design functionality by randomly replacing the respective original design gates. Further, the security analysis shows that our GateLock prevents SAT attacks and tremendously increases the brute force attempts over the existing techniques. The experimental evaluation of the proposed GateLock on ISCAS and ITC benchmark circuits shows that our method prevents the SAT attack completely while requiring only 1.43%, 1.08%, and 3.6% are power and delay overhead for embedding 128 keys.

This paper uses LUT-based tamper-proof that requires a slightly higher implementation cost than the normal memory; thus, it may increase the implementation cost of the proposed GateLock method. Therefore, our future work will explore the cost-effective way to store correct key sequences and provide an efficient implementation of the proposed method.

REFERENCES

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

[2] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[3] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, pp. 6:1–6:23, 2016.

[4] "International chamber of commerce, "impacts of counterfeiting and piracy to reach us $1.7 trillion by 2015,"." [Online]. Available: https://iccwbo.org/news-publications/policies-reports/economic-impacts-counterfeiting-piracy-report-prepared-bascap-inta/.

[5] SEMI, "Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to IP infringement." [Online]. Available: www.semi.org/en/Press/P043775, 2008.

[6] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 112–125, 2012.

[7] V. S. Rathor, B. Garg, and G. K. Sharma, "A novel low complexity logic encryption technique for design-for-trust," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 3, pp. 688–699, 2020.

[8] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 0030–38, 2010.

[9] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2012, pp. 953–958.

[10] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, 2016.

[11] K. Juretus and I. Savidis, "Reducing logic encryption overhead through gate level key insertion," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 1714–1717.

[12] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware IP protection," in *Proc. 51st Design Automation Conference (DAC)*, 2014, pp. 1–5.

[13] V. S. Rathor, B. Garg, and G. K. Sharma, "New light weight threshold voltage defined camouflaged gates for trustworthy designs," *Journal of Electronic Testing: Theory and Application*, vol. 33, no. 5, pp. 657–668, Oct 2017.

[14] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *Proc. IEEE 20th International On-Line Testing Symposium (IOLTS)*, 2014, pp. 49–54.

[15] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[16] K. Juretus and I. Savidis, "Reduced overhead gate level logic encryption," in *Proc. 26th edition on Great Lakes Symposium on VLSI*, 2016, pp. 15–20.

[17] B. Liu and B. Wang, "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks," in *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2014, p. 243.

[18] S. Khaleghi, K. Da Zhao, and W. Rao, "IC piracy prevention via design withholding and entanglement," in *Proc. 20th Asia and South Pacific Design Automation Conference*, 2015, pp. 821–826.

[19] V. S. Rathor and G. K. Sharma, "A lightweight robust logic locking technique to thwart sensitization and cone-based attacks," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 811–822, 2021.

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

[21] M. El Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes."

in *Proc. 22nd Annual Network and Distributed System Security Symposium*, 2015, pp. 1–14.

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

[23] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 127–146.

[24] ——, "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.

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

[26] ——, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[27] K. Shamsi, M. Li, T. Meade, Z. Zhao, Y. Jin, and D. Pan, "Appsat: Approximately deobfuscating integrated circuits," in *Proc. IEEE Symp. Hardware-Oriented Security and Trust*, 2017, pp. 95–100.

[28] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks," in *Proc. International Conference on Cryptographic Hardware and Embedded Systems*, 2017, pp. 189–210.

[29] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. Rajendran, "What to lock? functional and parametric locking," in *Proc. Great Lakes Symposium on VLSI*, 2017, pp. 351–356.

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

[31] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "Cas-lock: A security-corruptibility trade-off resilient logic locking scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 175–202, 2020.

[32] Y. Liu, M. Zuzak, Y. Xie, A. Chakraborty, and A. Srivastava, "Strong anti-sat: Secure and effective logic locking," in *Proc. 21st International Symposium on Quality Electronic Design (ISQED)*, 2020, pp. 199–205.

[33] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2514–2527, 2020.

[34] A. Sengupta, N. Limaye, and O. Sinanoglu, "Breaking cas-lock and its variants by exploiting structural traces," in *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021.

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

[36] Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou, "Sigattack: New high-level sat-based attack on logic encryptions," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 940–943.

[37] Y. Shen, A. Rezaei, and H. Zhou, "Sat-based bit-flipping attack on logic encryptions," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 629–632.

[38] V. S. Rathor, B. Garg, and G. Sharma, "New lightweight anti-sat block design and obfuscation technique to thwart removal attack," *Integration*, vol. 75, pp. 178–188, 2020.

[39] F. Yang, M. Tang, and O. Sinanoglu, "Stripped functionality logic locking with hamming distance-based restore unit (sfll-hd)–unlocked," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2778–2786, 2019.

[40] K. Juretus and I. Savidis, "Increased output corruption and structural attack resilience for sat attack secure logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 1, pp. 38–51, 2021.

[41] N. Limaye, E. Kalligeros, N. Karousos, I. G. Karybali, and O. Sinanoglu, "Thwarting all logic locking attacks: Dishonest oracle with truly random logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1740–1753, 2021.

[42] J. Zhou and X. Zhang, "Generalized sat-attack-resistant logic locking," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2581–2592, 2021.

[43] Q.-L. Nguyen, S. Dupuis, M.-L. Flottes, and B. Rouzeyre, "Skg-lock+: A provably secure logic locking schemecreating significant output corruption," *Electronics*, vol. 11, no. 23, p. 3906, 2022.

[44] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744–759, 2022.

[45] S. Patnaik, N. Limaye, and O. Sinanoglu, "Hide and seek: Seeking the (un)-hidden key in provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3290–3305, 2022.

[46] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 709–720.

[47] R. Karmakar, N. Prasad, S. Chattopadhyay, R. Kapur, and I. Sengupta, "A new logic encryption strategy ensuring key interdependency," in *Proc. International Conference on VLSI Design (VLSID)*, 2017, pp. 429–434.

[48] "(2011) nangate freepdk45 open cell library. nangate inc, 2011." in *[Online]. Available: http://www.nangate.com/?pageid=2325/*, 2011.

Vijaypal Singh Rathor is currently working as an Assistant Professor in the Department of CSE at PDPM Indian Institute of Information Technology, Design and Manufacturing (IITDM), Jabalpur, India, since August 2021. His research interest includes Hardware Security, Machine Learning and IoT, and Cloud Computing. He is also a member of IEEE since 2017.

Munesh Singh is currently working as an Assistant Professor in PDPM Indian Institute of Information Technology, Design and Manufacturing (IIITDM), Jabalpur, India. He has published many research articles in IEEE, Springer, and Elsevier journals. His research interests include Sensor Networks, Cooperative Computing, Radar Surveillance, Cyber Security, Machine Learning, and Intelligent Robotics.

Kshira Sagar Sahoo is currently acting as a post-doctoral fellow at the dept of computing science, Umea University, Umea, Sweden. He has more than 70 research articles in top tier journals and conferences. His research interests are SDN, Edge Computing, IoT, ML and Distibuted Computing. He is a senior member of IEEE and IETE.

Saraju P. Mohanty (Senior Member, IEEE) received the bachelor's degree (Honors) in electrical engineering from the Orissa University of Agriculture and Technology, Bhubaneswar, in 1995, the master's degree in Systems Science and Automation from the Indian Institute of Science, Bengaluru, in 1999, and the Ph.D. degree in Computer Science and Engineering from the University of South Florida, Tampa, in 2003. He is a Professor with the University of North Texas. His research is in "Smart Electronic Systems" which has been funded by National Science Foundations (NSF), Semiconductor Research Corporation (SRC), U.S. Air Force, IUSSTF, and Mission Innovation. He has authored 450 research articles, 5 books, and invented 10 granted/pending patents. His Google Scholar h-index is 56 and i10-index is 235 with 13,000 citations. He is a recipient of 17 best paper awards, Fulbright Specialist Award in 2021, IEEE Consumer Electronics Society Outstanding Service Award in 2020, IEEE-CS-TCVLSI Distinguished Leadership Award in 2018, and the PROSE Award for Best Textbook in Physical Sciences and Mathematics category in 2016.