Noise-based logic locking scheme against signal probability skew analysis

Ahmad Rezaei  |  Ali Mahani

Department of Electrical Engineering, Reliable & Smart Systems Laboratory, Shahid Bahonar University of Kerman, Kerman, Iran

Correspondence
Ali Mahani, Reliable & Smart Systems Laboratory, Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran. Email: amahani@uk.ac.ir

[Correction added on 8 May 2021, after online publication. The correspondence email address is updated.]

Abstract
Due to integrated circuit (IC) production chain globalisation, several new threats such as hardware trojans, counterfeiting and overproduction are threatening the IC industry. So logic locking is deployed to hinder these security threats. In this technique, an IC is locked, and its functionality is retrieved when the right key is loaded onto it. We propose ‘noise-based’ logic locking, consisting of two separate compliment blocks, which function in three states. By flipping a signal once in the circuit, these modules add corruption to the circuit, whereas either flipping the same signal twice or not flipping leads to the correct functionality. Thus, a low probability skew with a low corruption in the output is obtained by utilisation of these flipping states. We have improved SAT attack resiliency based on time by 17% for a locking block with 14 primary inputs in comparison with the well-known anti-SAT. The area overhead is less in comparison with other schemes, in which extra dummy parts or obfuscation elements are added to their circuit. Also, more crucially, our locking blocks are immune to SPS attack solely. After executing various attacks, retrieved circuits indicate improved overall resiliency against automatic test pattern generation based and approximate guided removal attacks as well.

1 | INTRODUCTION
Primarily, integrated circuit (IC) design, fabrication, and test were all processed in a single facility, and trusted foundries handled all the procedures. Recently, however, technological advancements, determinant time-to-market, and high facility setup and upgrading cost have forced the IC companies to manufacture their ICs in the third-party foundries off-shore. This multi-vendor cooperation is liable to multiple security threats. Hence, the semiconductor industry is posed to these adverse effects and loses about 4$ billion annually [1].

These threats can be executed by a malicious foundry or end-user to gain control over exact netlist of valuable IPs through IP theft or reverse engineering [2]. The objective of attacker varies, as an attacker can use IP in overproduction and illegal cloning (counterfeiting) [3] to sell in black markets. At the same time, this control over IC leads to modification of the circuit and insertion of hardware trojans [4] aiming at the malicious functional usage in next steps of IC application.

Furthermore, these threats can lead to information exploitation if reverse engineering is deployed. Henceforth, several retreat methods such as watermarking [5], split manufacturing [6], IC camouflaging [7] and logic locking [8, 9] were proposed to withstand IP piracy. The designer solely devises logic locking, and its exact function is out of reach for other third parties, while other methods, for instance, camouflaging target protection from end-user, and offer no protection from foundries. Therefore, logic locking has gained interests due to this superiority, and its design phase feasibility and efficiency.

Due to the aforementioned superiorities, logic locking is the focus of this study. In this method, some locking elements are coupled with the main circuit, which is controlled with key inputs (KIs) receiving from an on-board tamper proof memory, and circuit functions correctly just when the correct key bits are received as shown in Figure 1. There are a variety of locking methods such as random logic encryption (RLE) [8], strong logic encryption (SLE) [10], anti-SAT [11], SARILock [12], SLL [13], TT-Lock [14] and PLG-based logic locking [15],
and although various researches explored for a novel locking technique that hinders all possible attacks, to the best of our knowledge each method has drawbacks in the form of area and power overhead, and the vulnerabilities against different attacks. To evaluate the proposed methods, possible attack scenarios are discovered, and these fall into two main subcategories of algorithmic analysis attacks and structural analysis attacks; for instance, satisfiability checking based (SAT) attack [16] and automatic test pattern generation (ATPG) [10] techniques deal with the locking scheme’s algorithmic weak points, whereas in probabilistic attacks [17] and bypass attacks [18] the structural characteristics are addressed.

The circuit is locked with a logic locking technique which makes it bullet-proof against structural analysis attacks, which are underlain by structural traces in the circuit. At the same time, our locking method is adequately resilient against sensitisation and SAT attacks, which are both different types of algorithmic analysis attacks. In this regard, although other researchers have explored various methods to hinder attacks above, their methods have various drawbacks. These are indicated as limitations in the form of high area overhead (introduced by extra dummy parts), a finite number of protected patterns after the execution of SPS attack, and higher area overhead stemmed from resynthesisisation procedure. We have designed a locking block with an untraceable output by SPS attack; this locking method fulfils mentioned limitations and vulnerabilities by protecting all the input patterns even after SPS execution. Our proposed method adds lower overhead, as no extra part other than locking blocks are introduced to the circuit.

Rest of the study is contributed as follows:

1. Logic locking’s background is elaborated to discuss drawbacks of the former methods and the characteristics that can be improved.
2. We propose our noise-based locking technique, furthermore, to discuss the design and analytical aspects of the proposed technique.
3. Security analysis on the proposed technique from both aspects of signal probability and SAT resilience is highlighted.
4. Several simulations are designed to validate the lock’s security and results are assessed and discussed.
5. Eventually, our research results are addressed, and the study is concluded.

2 | PRIOR WORKS

Logic locking methods have evolved from different aspects through recent years, by the introduction of new attacks, and it is crucial to know this development further to design and improve the current techniques.

Logic locking methods are generally divided into three categories, as follow:

2.1 | Traditional methods

The initial methodology about logic locking was proposed in EPIC [8]. Its prime aim was to change the circuit’s functionality to reach the highest corruption in functionality for each wrong key (P) through the insertion of different locking elements like logical gates [10, 8], lookup tables (LUTs) [19], and multiplexers [20]. These insertion sites were random as in RLE [8], so they were subjected to simple ATPG or sensitisation techniques [10], which work with the propagation of key-value to the corresponding outputs to retrieve the correct key.

Subsequently, a new locking method was also in [10] devised to overcome this weakness. SLE, which benefits from reconvergent locking gates that by creating a chain of key gates, rely on each other, form a clique, and this chain of key gates can only be unlocked with brute force. In other words, logic gates were added in exact points to form the largest clique, which bound up finding a value of a KI on knowing other KIs’ value, and further, the attacker must try all patterns to unlock the locking system.

2.2 | SAT resilient methods

In 2015, the proposed SAT attack [16] unlocked all high corruption locking methods in a relatively shorter amount of time.
and commenced a new era in locking techniques. This attack heavily relies on the output corruption and as the attack proceeds by the elimination of wrong keys, adding corruption, in each iteration, key set area shrinks. This elimination finally leads to the exploitation of the remaining correct keys. Thus, while the mentioned logic locking techniques followed the maximum corruptibility approach, they were prone to be unlocked in a few iterations.

To thwart this problem, SAT resilient logic locking methods were proposed [11, 12] (Figure 2a,b), following the idea of using an embedded locking blocks with low corruption ratio. In such schemes, SAT attack needs an exponential number of efforts to obtain the right key. As shown in Figure 2a, anti-SAT consists of two functions (F and F-bar), which have $2n$ inputs ($2n$ KIs XORed with $n$ inputs from the main circuit). These functions are Andered/Ored, based on anti-sat block type, to form a low corruption block. This method was successful against SAT solvers; however, owing to the direct correlation between low output corruption and SAT attack iterations in this approach, these low corruption methods were found vulnerable to another type of attack.

2.3 Removal attack era

Anti_SAT block fulfills algorithmic weaknesses, but a much feasible attack called SPS [21] used structural drawback of Anti_SAT blocks and retrieved main circuit’s function in a relatively shorter amount of time in comparison with SAT attack. This attack uses the probability of nodes in a circuit to obtain the point which locking block is connected to the main part. While in Anti_SAT proposed method probability of this point is skewed towards 0 or 1, no matter how large the circuit is, the connection point is obtained in a matter of seconds. Consequently, other locking methods like SAR-lock (Figure 2b) were found vulnerable to this method, as they had the same attitude towards SAT attack.

To the best of our knowledge there are several new methods including AND Tree Insertion (ATI) [22], Camo-perturbation [23], design withholding and wire entanglement [19], TT-Lock [14], SFLL-hd [24] and approximate Anti_SAT [25] are devised to mitigate this problem. These methods are deemed to strengthen three aspects of circuits protection. These aspects are elaborated in the following, and their limitations and vulnerabilities are discussed:

Figure 2 (a) Original circuit locked by n input Anti_SAT block and (b) original circuit locked by n input SARLock scheme.
2.3.1 | Design obfuscation

In these methods, several dummy blocks, which are identical to locking blocks, are added to the circuit. This insertion raises the number of low probability sites in the circuit. Hence, after running SPS attack, the attacker ends up having more than one candidate for the locking block’s output. Also, design withholding and wire entanglement are used towards adding complexity to the circuit and impeding with detection of the locking circuit by replacing some parts with LUT blocks and insertion of multiple spare wires and multiplexer units. Both of these procedures leave an extreme impact on area overhead of the circuit, as extra dummy parts are added to the circuit.

2.3.2 | Modification of circuit

These methods modify the circuit’s output for one protected cube. Next, after a resynthesis process, a locking block is added in a way to restore the right output in case of setting the right KIs. Although in this method removing the locking block leads to a modified circuit, and this modified circuit is conceived not to be the correct answer of the original circuit. These methods are still vulnerable to probabilistic attacks, as the resorted circuit practically is almost identical to the real circuit except for some of the input patterns. Furthermore, in a real-world situation, the foundry can set aside this faulty input pattern and use the modified circuit or use existing bypass attacks. Also, a cutting edge attack is proposed in [26], which unlocks SFLL-hd method in order of minutes.

Also, the approximate Anti_SAT method [25] acts the same as mentioned schemes, and it also makes use of an adjustable corruption ratio in the output.

Both types suffer from overhead introduced in resynthesis process, for example, in the approximate Anti_SAT scheme three DTL blocks are utilised to make the circuit SPS proof. In contrast, in this scheme, for Anti-SAT-DTL locking block, only two DTL blocks are needed to lock the circuit with no resynthesis procedure.

2.3.3 | PLG-based method

Recently, in [15], a locking method based on gates built by silicon nanowire FET technology was proposed. In this technology, the type of gates varies based on their the property of controlling KIs. Thus, different configurations are implementable with the same structure.

In this method, a proportion of gates in the main circuit are exchanged with SiNW gates with varied functionality. Next, a tree of AND/NAND or AND/OR blocks, called RSAT, is coupled with the circuit through a single XOR/XNOR gate. Finally, replaced gates in the main circuit share same KIs with RSAT block, and a random wire from the circuit is used to obfuscate the locks’ outputs.

The drawbacks of this method are as follow. First, this method proposes an extra 6.25% delay to the circuit, which is much higher than 0.6% on average in the TT-Lock method. Next and more importantly, this method like design obfuscation methods replaces gates with new elements (PLG-type gates) which leads to higher area overhead.

Additionally, the modification procedure is more complex as it contains several steps in contrary to former methods like Anti_SAT.

All in all, these methods all have their respective drawbacks, and a fourth aspect, which is finding a locking block with the low probability skew and low corruption in the output, should be taken into consideration. This aspect was priorly examined using a daisy chain structure to lock the circuit in CAS-Lock [27] method. But it was afterwards found impractical in [28], as this method was unlocked without the need of an oracle circuit.

We explore this fourth aspect of SPS resiliency and propose a novel logic locking technique. Our technique benefits from two coordinated blocks with adjustable probability skew in the outputs and functions in such a way, which are neither found with SPS attack nor feasible for SAT attack to obtain the correct KIs. This approach consumes less area than the obfuscation methods, and if we remove the point with the highest skew in the locking circuit, it produces a full corrupted circuit in the output for all patterns rather than leading to a low corruption modified circuit.

3 | ATTACK ELABORATION

An ideal logic locking method has to hinder three main attacks; sensitisation attack, SAT attack and SPS attack. The sensitisation attack is not mentioned in this section, and it is discussed in security analysis in the next section. The other two attacks with a secondary attack called AGR attack are elaborated in the follow-up.

3.1 | SAT attack

In this section, execution requirements to deploy SAT attack by the malicious foundry are mentioned, and an overview to the attack methodology and its algorithm is elaborated in the following.

3.1.1 | Requirements

In an unreliable foundry, which in the IC is created, a locked netlist of the main circuit can be acquired in a conjunctive normal form (CNF). This CNF contains the relation between inputs, KIs and outputs and is the first imperative to retrieve the main circuit’s functionality. The second item needed in this procedure is a functioning IC bought from the market; this IC is used to evaluate the CNF to exploit the correct key patterns.
3.1.2 | Overview

SAT attack is an iterative attack, which through iterations various patterns are applied to inputs and KIs of the CNF function. Then on, by identifying the distinguishing patterns (DPs) in the output, these patterns are compared to the functional IC’s output and the key search area is discarded of the wrong key patterns. Thus, the attack proceeds till all the DPs are eliminated from the key set, and only the correct key patterns remain.

3.1.3 | Algorithm

Aforementioned CNF functions as \( O_p = F_c(I_n, K_m) \) [11] form, while the registered IC has the formula in the shape of \( O_p = F_m(I_n) \). I, K, and O respectively stand for inputs, KIs and outputs, and F corresponds to the relation between these three elements, c, and m consecutively stand for the corrupted and main circuit. SAT attack in each iteration applies a pattern from input set to CNF formula and after detecting DPs, the incorrect key patterns are rolled out of key set with the help of functioning IC, for instance, if \( I_n \) is set to \( I_i \) and \( K_m \) is set to \( [K_o \& K_1] \in K_m \), two possibilities are likely, first, the responding \( O_k \) of both formulas is identical \( [O_0 = F_c(I_i, K_0)] = [O_1 = F_c(I_i, K_1)] \), in this case the CNF reveals no DP, so the key patterns are randomly changed to find a DP, secondly the responding outputs, \( O_0 \) and \( O_1 \), are not identical, \( [O_0 = F_c(I_i, K_0)] \neq [O_1 = F_c(I_i, K_1)] \) reveals a DP in CNF formula, if so, the wrong key is eliminated from key set \( K_m \). Subsequently, this process proceeds for several iterations to narrow down the key search area to the correct keys, which have identical responses same as functional IC for all the input patterns. The corresponding algorithm is given in Algorithm 1.

Algorithm 1 SAT attack algorithm [16]

1: Input CNF, ICoracle
2: Output \( \overline{R_{correct}} \)
3: \( i \leftarrow 0 \)
4: \( F_c: = F_c(I_i, K_1) \& F_c(I_i, K_2) \quad [O_i \neq O_j] \)
5: while all DPs are not discovered do
6: \( I_i: = sat\_assignment(F_c) \)
7: \( Wrong\_keys\_check\_oracle(ICoracle, F_{c1}, F_{c2}) \)
8: \( G_{i+1}: = G_i \& CNF(I_i, K, O_i) \)
9: \( F_{i+1}: = omitted\_wrong\_keys\_in\_F_i \)
10: \( i: = i+1 \)
11: endwhile
12: \( \overline{R_{correct}}: = sat\_assignment[O_i] \)

3.2 | SPS attack

As mentioned in Section 2, most of the cutting edge locking techniques were found vulnerable to removal attacks, with SPS being the basis of these attacks. Thus, SPS attack is of value to discuss. In the following, we address the necessities to execute SPS attack, an overview, and elaboration of the SPS algorithm flow.

3.2.1 | Requirements

In this attack strategy, a locked netlist of the circuit in an unreliable foundry or reverse engineered netlist solely can be used to execute the attack on the circuit.

3.2.2 | Overview

Anti_SAT block and other schemes, which are designed to thwart the SAT attack, have low corruptibility in the output, and as it is obvious from Figure 3 [14] that SAT attack resilience and SPS attack resilience have inverse correlation with respect to the \( P_c \) value (corruption per key). This inverse correlation is due to the fact that achieving best resiliency for a SAT resilient blocks rests upon low corruption injection to the circuit, and prior locking schemes like Anti SAT and SAR Lock followed this strategy to overcome the SAT attack. On the other hand, this low corruption means a low activity in the output gate of these locks (as they used AND-XOR combination to merge and couple the lock with the circuit). This low activity \( (P_c \) close to \( 2^0 \) or 1) means a higher probability skew from 0.5 that leads to the detection by the SPS attack. Also, to fulfill the SPS resiliency higher activity in the output gate of the prior blocks, and as a result of this higher activity, higher corruption is needed. This higher corruption further leads to early termination of SAT attack on the corresponding circuit, while more key values are omitted from the key set area in each iteration of SAT attack due to \( P_c \) close to \( 2^0/2 \).

These locking methods leave a trace in the form of high probability skew from 0, which is considered as the reference probability. Moreover, by obtaining the probability of each wire in the given netlist, the highest signal probability skew is identified, which ravel the wire which connects the locking block to the main circuit. Then, this wire is disconnected from the circuit, and the main circuit’s functionality is restored.

3.2.3 | Algorithm

Signal probability skew is obtained from Equation (1), and it defines the probability of making the right guess about the value of a signal. This probability of making a right guess is in range of \([-0.5, 0.5]\) and when \( s \) has the value 0, the chance to guess its value is negligible, while this probability of making the
right guess culminates when $s$ has value next to its boundary limits ($-0.5, 0.5$).

$$s = P_r[x = 1] - 0.5 \quad (1)$$

Different gates have different impacts on the signal probability skew; for instance, XOR brings the probability close to 0, while AND gate pushes the probability skew onward $-0.5$; these effects can be obtained from Equations (2)–(5). In fact, SPS algorithm starts from the inputs and benefits from these formulas to obtain the signal skew on each wire, as inputs are set to 0 skew, while other wires’ skew will be calculated. This computation is decisively fast, and the accuracy improves as the size of the circuit enlarges.

$$S_{\text{AND}} = 0.5 \times (s_1 + s_2) + s_1 \times s_2 - 0.25 \quad (2)$$

$$S_{\text{OR}} = 0.25 + 0.5 \times (s_1 + s_2) - s_1 \times s_2 \quad (3)$$

$$S_{\text{XOR}} = -2s_1 \times s_2 \quad (4)$$

$$S_{\text{NOT}} = -s_1 \quad (5)$$

Eventually, in this case, as SPS attack is solely targeted in this section, the wire with the closest skew to -0.5 or 0.5, and its transitive fan-in (TFI) will be eliminated from the circuit to attain the circuit’s functionality. SPS attack’s algorithm is shown in Algorithm 2.

### Algorithm 2 SPS analysis algorithm

1. **Input** Netlist
2. **Output** Candidate list
3. **Initialise** input_wires_skew ← 0.5
4. $Skew\_\text{array} ← []$
5. **for** all gates in netlist **do**
6. $Skew\_\text{array}(g_i) ← \text{compute}_\text{Skew}(g_i)$
7. **endfor**
8. **Sort** values($Skew\_\text{array}$, Candidate list)
9. Maximum_Skew ← Candidate list(1)
10. **Remove** TFI(Netlist, Maximum_Skew)

### 3.3 Approximate guided removal (AGR) attack

Approximate guided removal (AGR) attack, as discussed in [17], is a subsequent attack mainly based on SAT attack kernel and structural analysis. This attack takes less time than the other attacks and works very well against low corruption schemes to retrieve the original circuit. Hence, it is worthy to elaborate on this attack as well. In the follow-up, an overview of the attack and its initial requirements is discussed, and next, AGR attack’s algorithm is presented.

#### 3.3.1 Requirements

This attack needs three initial requirements. First, locked or reverse engineered netlist is the same as the one in the SPS attack. Next, functioning IC bought from the market; this IC is used as an oracle to exploit the correct key patterns. Finally, the number of KIs ($n$) corresponding to the low corruption scheme.

#### 3.3.2 Overview

AGR attack is considered as a secondary attack. This means that attackers examine this attack on the circuit only when sensitisation is terminated unsuccessfully, SAT attack leads to long trials driven by low corruption in the circuit, and the original circuit is not retrieved after the execution of SPS attack.

As mentioned earlier this attack is devised based on two different kernels SAT attack and structural analysis, which are used in one of its two stages:

1. Utilising approximate SAT (AppSAT) attack to peel off traditional locking methods from low corruption schemes
2. Structural tracing and output gate removal

In the first stage, an AppSAT attack is used to remove traditional locking schemes from low corruption blocks. The
objective in this stage is to make use of SAT attack feasibility against high corruption methods to peel off security level. Then, in the second stage, the circuit with a degraded security level, lock block is omitted from the main circuit by following the transitive fan-outs (TFOs) of KIs. When all wires converge to a single wire, the convergent wire is assumed to be the output wire of the locking block.

Under an ideal assumption, after the execution of the first stage, only the KIs corresponding to the low corruption scheme remain, and all traditional KIs are peeled off. But, in a real setting, AppSAT attack only distinguishes traditional scheme from low corruption scheme partially. Hence, only fluctuating key bits are used in the second stage, while stable keys could belong to each scheme.

Moreover, TFOs of different proportions of these fluctuating KIs are followed to their corresponding convergent gates. These convergent gates are sorted for various proportions of these KIs and are ranked regarding the number of associated KIs. Finally, the convergent gate which has the higher rank is regarded as the locking blocks’ output and its TFI are removed to retrieve the original circuit.

3.3.3 | Algorithm

A pseudo-code of AGR attack algorithm is demonstrated in Algorithm 3. This algorithm is explained as follows.

First, AppSAT attack is executed several times on the circuit. In each execution and after terminating stable KIs, convergent gates \((g_i)\) with TFI of fluctuating KIs are obtained. These \(g_i\) are added to candidates list if KIs in the TFI \((C_{gi})\) are close to \(4n\) and \(R_1(g_i)\) and \(R_2(g_i)\) values are approximately equal to 0.5. \(R_1(g_i)\) and \(R_2(g_i)\) are division of corresponding number of two fluctuating key input into total number of fluctuating KIs in \(g_i\). These divisions are shown in Equation (6).

\[
R_1 = \frac{C_{in1}}{C_{gi}}, R_2 = \frac{C_{in2}}{C_{gi}}
\]

where \(C_x\) stands for the number of fluctuating KIs that converge at a given gate \((x)\). Moreover, \(R_1(g_i)\) and \(R_2(g_i)\) variables are assumed to be close to 0.5 because the majority of low corruption schemes are built upon two blocks with the same number of inputs \((2n)\). Also, in most of the low corruption schemes, if \(g_i\) is locking the output of the blocks, it should have all \(4n\) inputs in its TFI.

Next, we sort the candidates based on the number of convergent KIs. Among these sorted candidates, the one with the maximum number of convergent KIs is considered as locking block’s output gate \((G)\). Finally, TFIs of gate \(G\) are removed from the locked netlist, and the original circuit is retrieved if \(G\) is found correctly.

Algorithm 3 AGR attack algorithm [17]

| Line | Description |
|------|-------------|
| 1:   | Input Netlist |
| 2:   | Input \(n\) |
| 3:   | Output Retrieved Netlist |
| 4:   | Initialise \#cand = num_gates(Netlist) |
| 5:   | while ( \#cand > 1 and !timeout ) do |
| 6:   | launch appsat(4); // make 4 appSAT calls |
| 7:   | candidates = {} |
| 8:   | for ( \(g_i\) \in Netlist ) do |
| 9:   | if \(C_{g_i} \approx 4n\) and \(R_1(g_i) \approx R_2(g_i) \approx 0.5\) then |
| 10:  | add \(g_i\) to candidates |
| 11:  | \(G \rightarrow \) find maximum_key_count(candidates) |
| 12:  | Retrieved_Netlist = remove_TFI (Netlist, \(G\)) |

However, the removal of resilient methods above introduce overhead by having obfuscation, and extra parts added to locking block, or they protect a finite number of patterns after implying SPS attack. We have devised a locking technique, in which the locking block solely protects all the input patterns from these main attacks, and it does not introduce overhead in the form of extra added parts.

4 | PROPOSED METHOD

In this section, our noise-based lock’s design methodology is discussed, and afterwards, security aspects of the proposed locking method are elaborated.

4.1 | Noise-based logic locking

SAT attack’s execution time is obtained from: \(T = \sum_{i=1}^{\lambda} t_i\), where \(\lambda\) is the number of iterations and \(t_i\) stands for time elapsed in the \(i\)th iteration [11]. Therefore, SAT resiliency is achievable by increasing one of these two factors, and \(T\) is at its apex when just one wrong key is eliminated in each iteration.

The ideal relation between KIs, circuit’s inputs, and generated corruption in the locking block’s output (for \(KI = 3\)) is illustrated in Table 1, as it takes an exponential amount of time for SAT solvers to unlock the circuit locked with this low corruption characteristic. This scheme is almost identical in all SAT resilient blocks, and due to structural traces, former implementations of this truth table were unable to resolve structural traces in their low corruption output, which leads to SPS vulnerability. Henceforth, we propose a locking method to fulfil this need for the follow-up.

In our method, as is indicated in Figure 4, for \(na\) input locking block, two functional parts are coupled together with \(n\) inputs from the primary inputs and \(2n\) KIs. These two parts are connected to the same inputs of the main circuit, but the
KIs’ output in each block differs. Each block functions as one of truth tables in Table 2 and the corruption is injected into main circuit’s signal next to its outputs.

Furthermore, these two separate blocks are allocated in a way that together when KIs in both blocks are equal and act the same as a low corruption scheme, while for other patterns of KI, the corruption is varied as can be illustrated in Figure 4, the locking block consists of two functional units connected to a wire with lower observability of 30% (secure integration of the block [11]) from the main circuit, through two separate XOR gates. This structure imposes the difference between truth tables (a) and (b) in Table 2 to the main circuit’s functionality ($F_{m}$). Moreover, this locking technique for the right key and other non-corrupted patterns inverts the signal value twice or not at all, which means the functionality of the main circuit experiences no change. Whereas in corrupted patterns, the value of XOR gates varies, so the signal value will be inverted once, and the corruption is inserted to the circuit’s functionality. These truth tables have a lower probability skew from 0.5 to impede with the SPS attack succeed ratio. These blocks are not identical, and the only crucial factor is having T1 for equal KI values as a result of XORing of locking blocks.

Primarily, in other locking schemes, KIs are XORed with PI_s and used as locking block’s inputs. These XOR gates add extra overhead to the circuit, while XOR has the highest area consumption among the logic gates. Thus, we have designed both blocks in a way to function without adding XOR gates. In other words, KIs and PI_s are locking block’s direct inputs, which decrease the area overhead.

As demonstrated in this section, our locking method introduces no overhead other than the locking blocks, while in the obfuscation methods, several extra parts, such as LUTs, multiplexers, and dummy trees, are added to hide the locking block’s output during the SPS analysis attack. Also, this method protects all the input patterns by not allowing the SPS attack to remove the locking block from the main circuit, which will be discussed afterwards.

| Table 1 | Anti_SAT truth table (for $n = 3$) |
|---------|----------------------------------|
| Anti_SAT | $k_0$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ | $k_6$ | $k_7$ |
| 000     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 001     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     |
| 010     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     |
| 011     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     |
| 100     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     |
| 101     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     |
| 110     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     |
| 111     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     |

| Table 2 | Noise-based lock block’s truth tables (for $n = 3$) |
|---------|----------------------------------|
| $F_0$   | $k_0$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ | $k_6$ | $k_7$ |
| 000     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 001     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     |
| 010     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     |
| 011     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     |
| 100     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     |
| 101     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     |
| 110     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     |
| 111     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     |

| $F_1$   | $k_0$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ | $k_6$ | $k_7$ |
| 000     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 001     | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     |
| 010     | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0     |
| 011     | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 0     |
| 100     | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 0     |
| 101     | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 0     |
| 110     | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 0     |
| 111     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 1     |

**Figure 4** Noise-based lock diagram
4.2 | Evaluation of the proposed method

We strongly believe that our noise-based locking method repels possible attacks. Henceforth, in this section, we will discuss the resilience of our proposed method against three major attacks, which are the basis of other attacks as well, and are also the well-known AGR attack.

4.2.1 | Sensitisation attack resilience

In this locking scheme, \( F_0 \) and \( F_1 \) pair blocks share the same inputs, and KIs all converge into flipping signals at the output of these blocks. In case of a sensitisation attack, all of the KIs should be set using brute force to propagate one of the KIs to blocks’ output; thus, using such attack on our proposed block is not feasible [14]. Moreover, the other positive point about this scheme is the concurrent impact of locking blocks on the main circuit, as the corruption is injected into the circuit through one of XOR/XNOR gates, but it is then filtered by the other one in many cases. Hence, to obtain the correct functionality, values of these outputs are varied based on the locking blocks’ functionality and also, the XOR/XNOR gate in the output of these blocks. Therefore, making it even harder to attain the key values with sensitisation attack.

4.2.2 | SAT attack resilience

The SAT attack can be repelled if all the DPs of the circuit are discovered in exponential time. Hence, the crucial aspect is to maximise the number of iterations to hinder this attack. For instance, in a given circuit with \( 2n \) KIs, DP recognition is directly underlain by the number of wrong keys in the main key set. First, we discuss Anti SAT’s involving factor in SAT attack resiliency, and then our method’s superiority is elaborated.

In Anti-SAT [11] method, the number of wrong keys detected per iteration is obtained from equation below.

\[
Wk_{iter} = P_c \times (2^n - P_c)
\]  

(7)

The number of KIs which output 1 is \( P_c \), which is a constant value for all the input patterns and the number of KIS that output 0 will be \( 2^n - P_c \). \( Wk_{iter} \) is obtained from Aended output of \( F_0 \) with \( F_1 \) in Anti_SAT block in Figure 2a [11]. Furthermore to detect all the DPs, several iterations (\( \lambda \)) are needed, which can be calculated by dividing the number of all the wrong KIs from Equation (7).

\[
\lambda \geq \frac{2^{2n} - 2^n}{P_c \times (2^n - P_c)}
\]  

(8)

In this equation, changing \( P_c \) within its range (1 ≤ \( P_c \) ≤ 2^n) leads into varied SAT attack resiliency shown in Figure 3. This shows that the best SAT attack resistance is obtained when \( P_c \) is set to 1 or 2^n.

In our proposed method, wrong keys are separated among two separated blocks, and the XOR of these truth tables reveals the functionality of the locking method. For a given \( 2n \)-sized key, \( n \) PIs are locked. This relation is discussed in Equation (9):

\[
O = F_0(\overline{X}_i, \overline{K}_i) \oplus F_1(\overline{X}_i, \overline{K}_i)
\]  

(9)

where \( F_0 \) and \( F_1 \) are functions indicating the KIs and PIs relation for individual block, also \( O \) states corruption in the output. To estimate iterations in our noise-based blocks, we need to justify the condition that DP found per iterations in Equation (9). A DP is detected when \( O = 1 \); therefore, the XOR of both blocks should be ‘1’.

\[
O = 1 = F_0(\overline{X}_i, \overline{K}_i) \oplus F_1(\overline{X}_i, \overline{K}_i)
\]  

\[
\Leftrightarrow [F_0(\overline{X}_i, \overline{K}_i) \wedge F_1(\overline{X}_i, \overline{K}_i)]
\]  

\[
\vee [F_0(\overline{X}_i, \overline{K}_i) \wedge F_1(\overline{X}_i, \overline{K}_i)] = 1
\]

In Equation (10), we have set the DP condition. In our method, as our blocks are not identical, we need to define two new variables; \( P_{0i} \) and \( P_{1i} \), which are the number of ones in each block (\( F \)) input pattern \( X_i \). In contrast with the Anti_SAT method these values are not constant, and vary in range 1 ≤ \( P_{0i} \) ≤ \( 2^n - 1 \) and 1 ≤ \( P_{1i} \) ≤ \( 2^n - 1 \) for different PIs.

\[
L_{0i} = \overline{L} \cdot F_0(\overline{L}) = 1, \quad (L_{0i} = P_{0i})
\]

\[
L_{0i} = \overline{L} \cdot F_0(\overline{L}) = 0, \quad (L_{0i} = 2^n - P_{0i})
\]

\[
L_{1i} = \overline{L} \cdot F_1(\overline{L}) = 1, \quad (L_{1i} = P_{1i})
\]

\[
L_{1i} = \overline{L} \cdot F_1(\overline{L}) = 0, \quad (L_{1i} = 2^n - P_{1i})
\]

As \( P_{0i} \) and \( P_{1i} \) are varied based on different KIs, we should specify our lock’s functionality and then, calculate number of DPs in the locking block. We specify our locking blocks to locking blocks with truth table resembling 2, expect our KI is set on \( 2n \) to lock \( n \) PIs (Figure 5).

In these truth tables, the total number of DPs are obtained from \( L_{0i}^1, L_{0i}^0, L_{1i}^1, L_{1i}^0 \) under the condition in Equation (10).

\[
DP_{\text{repetitive}} = \sum_{i=0}^{2^n-1} (P_{1i} \times (2^n - P_{0i}) + P_{0i} \times (2^n - P_{1i}))
\]

(12)

If we explore the truth tables from \( X_0 \) to \( X_{2^n-1} \), both equations will result in Equation (12) which is dependent on values of \( P_{0i} \) and \( P_{1i} \) and also, contains repetitive DPs. Therefore, to compute the total number of DPs without repetition, we simplify the above equation into Equation (13), by calculating DPs that are not repetitive from \( X_0 \) to \( X_{2^n-1} \).
\[
DP_i = \begin{cases} 
2 \times (2^n - 1) & X_0 \\
(2^n - 1) & X_1 \\
\sum_{i=2}^{2^n-1} 2 \times (2^n - i) & X_i 
\end{cases}
\]

\[\rightarrow DP_{total} = \sum_{i=0}^{2^n-1} DP_i = 2^{2n} - 1\]  

Equation (13) states that SAT attack in the circuit locked with noise-based method with truth tables in Fig. 5 can terminate successfully only if all the input patterns are searched for DPs, while \(DP_{total} = 2^{2n} - 1\) and \(\lambda = 2^n\) iterations are needed to find all these DPs. In other words, the minimum number of DPs per iteration are detected in the \(2^n - 1\)th row of the truth table, and this value is equal to \(2^n\). Thus, SAT attack iterates through all KIs to succeed. Also, the third part of Equation (13) demonstrates that by discovering more DPs in each iteration, the next iterations consist of fewer DPs, which makes these trails last longer than others.

Our input patterns are limited to \(n\) input patterns; therefore, SAT attack search area is upper bounded by \(2^n\) iterations. This may seem vulnerability, while Anti_SAT block’s iterations (with the same number of KIs) are lower bounded to this value, while another crucial factor is needed to bare into mind.

This crucial factor in the SAT attack is the number of DPs detected in each iteration. The total time of an SAT attack is in correlation with both \(\lambda\) and the time of each iteration \((t_i)\) [29]. As a result, the time consumption of SAT attack can be increased if it searches a larger key set area per iteration.

In our proposed method, DPs found are declined dramatically after each iteration, which results in a higher CPU time than the Anti_SAT method. Also, another effective aspect on SAT attack time consumption is that noise-based method’s key set is larger than the Anti_SAT method with respectively \(2^{2n} - 1\) and \(2^n - 2^n\) wrong keys.

To sum up, locking blocks are adjustable to any value which serves the best corruptibility and \(\lambda\) to hinder the SAT attack by allocating an exponential time to unlock the circuit.

4.2.3 | SPS analysis resilience

Figure 3 illustrates the correlation between the mentioned attacks for prior blocks, while our focus is to alter this correlation in such a way to have an SAT resilient block, which is also resistant against the SPS attack.

In this technique, two light blocks, which are not unique, are used to tackle high signal probability skew in blocks’ output wire, which connects locking blocks to the main circuit. These blocks inverse the value of the signal in the circuit and add corruption to the circuit as they inverse the given value once, while they leave the value unchanged if they inverse the signal twice or not at all. This functionality allows us to implement a locking block added with a noise like ones or zeros in the truth table (based on output XOR/ XNOR connection), conceding us the right to adjust the output signal probability skew on our desired value.

Thus, executing an SPS attack on noise-based method results in not finding lock’s outputs due to their lower allocated probability skew.

4.2.4 | AGR resilience

In the worst-case scenario, only KIs corresponding to our Noise-based method are remaining in the locked circuit, and all other traditional KIs are found with an Approximate SAT attack. In this situation, AGR attack executes the second stage which is following TFOs to the convergent wire. In this stage,
TFOs of Noise-based KIs converge at the wire in the main circuit as our separate locking blocks are connected to the circuit with the help of two XOR gates.

Comparing with Anti_SAT method, AGR attack detects Anti_SAT’s output, since this scheme is connected to the circuit through a single XOR/XNOR gate such as in Figure 6a. On the other hand, this multi-stage attack identifies a wire in the main circuit in our proposed method. A schematic of our proposed lock is illustrated in Figure 6b, in which the identified wire is clearly an SO in the main circuit. Furthermore, in Anti_SAT block, the high skew of the output guarantees the success in this scenario whereas our method leaves no probability traces in the output, and henceforth, the attacker cannot discern the wrong candidate gate from the right ones based on their probability skew.

AGR attack concurs various prior locking schemes as they were connected to the main circuit through only one XOR/XNOR gate, whereas our noise-based lock is connected to the circuit through two XOR/XNOR gates that lead to the exploration of convergent wire in the main circuit.

4.3 Case study

We explain an instance of a circuit locked with our proposed locking blocks with respect to the fact that locking circuits are not unique, and various locks can be devised with the same input and output signals’ skew.

We have designed two compliment blocks $F_0$ and $F_1$ for $n = 3$ as depicted in Figure 7a,b. Output signal probability of these blocks is, respectively, 0.56152 and 0.44922 for $F_0$ and $F_1$ blocks.

For simplicity, we have used the C17 benchmark (Figure 7c) and locked this circuit with our noise-based method. As this circuit is small, executing named secure integration technique is not possible. Hence, we randomly have allocated $N_3$ gate’s output as the locking point. We have randomly connected three of PIs of the circuit to relevant locking block inputs and rest of the inputs is used as KIs. The outputs of locking blocks are connected to the output of $N_3$ through two XOR gates as shown in Figure 7d.

The SPS analysis is executed on locked C17 circuit, and probability results are illustrated in the circuits in Figure 7. In this circuit outputs of the lock do not have the highest probability skew, and are not found by SPS analysis, which confirms our claim.

Furthermore, the highest probability skew is indicated in the $M_8$ gate’s output wire, with a probability of 0.21875. This probability skew is however smaller than the highest probability skew in the Anti_SAT block with $n = 3$, which leads to a probability of 0.10937 in its output. This comparison is also valid for other prior methods.

SAT attack execution on our circuit shows that $2^n - 1 = 7$ iterations are needed for our circuit to be unlocked, which is exponential for $n = 3$.

Eventually, if we trace TFOs of our locking blocks coupled with the AGR attack L1 gate output wire in the circuit is found as the convergence point, meaning a failure in AGR attack completion. While removal of L1 leads to removal of $N_3$ gate in the original circuit.

5 SIMULATION RESULTS

In this section, at first, SAT attack and SPS attack resilience of the proposed method are evaluated through various experiments. Then, resiliency against a more detailed removal attack, Approximate guided removal attack, which can be helpful to identify the output of the locking block are examined to validate our method. Eventually, the area, power and delay overhead of our proposed method are elaborated.

5.1 SAT resilience

Herein, the proposed locking block’s security is evaluated through the implementation of several SAT attacks, and different factors are assessed. The benchmarks are from ISCAS 85 [30] and MCNC from [16], and Lingeling SAT solver [31] is used as an attack simulator tool. The CPU time limit is set on 10 h [16] and experiments are further obtained using an Intel Core i7 CPU W3690 CPU, with 8 GB RAM, running at 2.40 GHz.

5.1.1 Corruption per key (Diff)

We define KIs with the same pattern eliminated in each iteration from the key set area as (Diff), which stands for the difference between truth tables. In our method, Diff is obtainable by observing difference of both truth tables. Table 3
elucidates the impact of Diff on a number of iterations needed by the SAT solver to decrypt the locked circuit. Implementation of a 16 KI locking block clearly states the importance of having Diff close to 1 to gain the maximum resistance against this attack, while a varied value of \( P_c \) impedes with SAT resistance in higher Diff values.

In other words, the Diff value decreases as the number of iteration needed by SAT attack dramatically in our simulations; also, it is distributed uniformly between all the input patterns. Therefore, Diff in each iteration is in correlation with the number of found DPs, and it can terminate the attack rapidly if its value is high. While in Anti_SAT method, \( P_c \) improves SAT resiliency both in high and low values.

### 5.1.2 | Key size (n)

While designing the locking blocks, Diff is set to 1; therefore, \( \lambda \) will be in its apex concerning Diff variable, and maximum iterations are needed for each module to be unlocked. Table 4 illustrates this exponential growth of search area based on n, as can be seen, SAT iterations are in order of \( 2^n \). Regarding this attack, the time needed to unlock the circuits is also in a direct correlation with the n given, and it varies in n fixed from 8 to 16, respectively, from 7.254 to more than 10 h.

Furthermore, to confirm our claim in Section 4.2.2 that our locking blocks need more time to be unlocked by SAT attack than Anti_SAT blocks, we have executed a termination time comparison with our noise-based method.

We have designed various Anti_SAT type 0 blocks, same as in [11] for different number of \( n \) with highest SAT attack resistance in \( (P = 1) \). These Anti_SAT locks and our Noise-based locking blocks are coupled with six different

---

**TABLE 4** SAT attack average results (Diff = 1)

| n   | 8   | 10  | 12  | 14  | 16   |
|-----|-----|-----|-----|-----|------|
| Iterations | 255 | 1023 | 4095 | 16383 | 65535 |
| Time (s) | 7.254 | 75.312 | 1014.67 | 16373.8 | TO |

---

**TABLE 3** C1355 circuit locked with our noise based lock (n = 16) and assessment of Diff impact on SAT attack resilience

| \( P \) | 1   | 16  | 33  | 65  | 129 | 1025 | 2049 | 32769 | 65533 |
|--------|-----|-----|-----|-----|-----|------|------|-------|-------|
| Iterations | 65535 | 4094 | 2046 | 1022 | 510 | 71  | 32  | 4     | 1     |
| Time (s) | 1366.2 | 320.644 | 69.3101 | 19.3281 | 1.37964 | 0.5614 | 0.0799 | 0.0373 |       |

---

**Figure 7** (a, b) \( F_0 \) and \( F_1 \) Noise-based blocks for \( n = 3 \) with indicated signal probabilities, (c) original C17 benchmark circuit with associated signal probabilities and (d) locked C17 circuit using noise-based lock with indicated signal probabilities
benchmarks. Next, the SAT attack using the Lingeling software was executed to unlock these locked circuits, and CPU time report of each attack was gathered. Eventually, the amount of increase in the CPU time required in unlocking our method in comparison with Anti_SAT in the same situation is compared, and the ratio of improvement is reported in Figure 8. This figure clearly states an obvious increase in CPU time (s) with overall 37.66% and 17.09%, respectively, for \( n = 8 \) and \( n = 14 \). This superiority is due to having only one correct KI, whereas Anti_SAT contains \( 2^n \) correct KIs, and also, our method is more time consuming to find DIPs in each iteration because of a linear decrease in the number of found DPs based on Equation (13).

5.1.3 | Coupled locking block with traditional schemes

Secure logic locking scheme is primarily used to outweigh the ATPG based attacks, and consequently, it is utilised to insert high corruption in SAT resilient schemes. We have coupled different SLL with 10% overhead with C1355 circuit locking with a different number of KIs from our proposed method (Figure 9).

The simulation results strongly define that on using SLL as a traditional method with low corruption schemes, leads to a more computationally intensive locking scheme for the SAT attack to retrieve the circuit. In such schemes, the time needed to unlock the circuit is multiplied, for example, SAT solver’s CPU time to unlock the 10% SLL coupled with our method is six times more than its original value to solve our method solely.

Also we have assessed our results in this section for DAC’12 traditional locking method with 10% overhead. As illustrated in Figure 9, results slightly differ from SLL locking method.

5.2 | SPS resilience

Herein, to show noise-based SPS resiliency as it is discussed in Section 4.2.3, by using non-identical parts in both pair blocks of the locking circuit, the output signal probability is set to a value close to 0 skew. Henceforth, a new variable must be defined to scrutinise this resiliency. This new variable is the noise percentage (\( N_e \)), added to locking blocks’ truth tables, which will be fully filtered after passing two XOR/XNOR gates in the output of the circuit, and only the functionality of a low corruption circuit will remain.

Furthermore, in all these simulations Diff is set to 1, which guarantees the highest SAT attack resistance, as in all prior locking techniques, the SAT resilience has a reverse correlation with SPS attack (3). Regarding these simulations, modules are evaluated through the method in [21]. Benchmarks in this part are from ISCAS 85 [30] and ISCAS 89 [32], which are examined to find the high skew probabilities (closest to 0 or 1) in such circuits.

In this section, we first evaluate our locking method based on different numbers of \( n \); afterwards, we define a new variable to assess the adjustable output skew of our method. Eventually, we conclude this section by studying high skew wires in our locking block, which are potentially found in SPS attack.

5.2.1 | SPS attack results analysis

In this section, we have executed SPS attack on several circuits locked with \( N_e = 50\% \) and \( n = 16, 64 \) locks. As the objective of this attack is to find the output of the locking block, results

\[ \text{FIGURE 8} \quad \text{CPU time increase of noise-based locking method in comparison with Anti_SAT} \]

\[ \text{FIGURE 9} \quad \text{Time consumption of SAT attack on noise-based logic locking mixed with traditional schemes} \]
are sorted from the highest skew ratio to the lowest. These results are shown in Tables 5 and 6, in which the candidate number of $F_{l0}$ and $F_{l1}$ is the priority for selecting outputs of two locking block pairs among other wires in the circuit. Consequently, these tables demonstrate that on average, these outputs are in the last 25.4% candidates of circuits. Hence, execution of SPS attack on these circuits is impractical for the attacker.

5.2.2 | Noise impact on the circuit ($N_z$)

As Table 5 ravels by designing the locking block with a $N_z = 50\%$, the signal probability skew in the output of these blocks will be close to 0, and the SPS resilience is obtained, whereas in former Anti SAT method this skew had the highest value in the edges. Also, $N_z$ value for locking the block with $n = 16$ is adjusted on different noise values (noise percentage: 50\%, 25\%, 12.5\%, 6.25\%, 3.125\% and 0\%), and results of SPS attack are shown in Figure 10. These results state the direct effect of $N_z$ on signal output skew in such a way that outputs of locking block have a higher probability skew as corresponding noise percentage is closer to 0\%, whereas 50\% noise percentage leads to minimum skew.

5.2.3 | Blocks’ structural analysis

In this technique, locking blocks are not identical, and multiple blocks with the same amount of noise injection can be designed with varying truth tables and circuit structures. Thus, here, we have elaborated on our functional blocks, which we have used in the simulation part. Already stated blocks have the same truth tables as in Table 2 with variable $n$. First, these circuits are designed using Verilog HDL language, and afterwards, they are optimised and implemented using Design Compiler software.

After execution of the SPS attack on these blocks, the probability skew of different wires in the blocks are obtained. However, the majority of these wires have a reasonable probability skew; in one of the blocks, a wire suffers from high probability skew. Henceforth, to further scrutinise the SPS attack, we have elaborated on the output skew of these blocks.

| Benchmark | Wire count | $F_{l0}$ Candidate number | $F_{l1}$ Candidate number |
|-----------|------------|---------------------------|---------------------------|
| C5315     | 2603       | 1793                      | 1794                      |
| C7552     | 3928       | 2921                      | 2922                      |
| S9234     | 6064       | 4738                      | 4739                      |
| S13207    | 8860       | 6656                      | 6661                      |
| S15850    | 10592      | 8592                      | 8597                      |
| S35932    | 18037      | 12443                     | 12444                     |
| S38417    | 24052      | 19986                     | 19988                     |
| S38584    | 20926      | 17976                     | 17977                     |

| Benchmark | Wire count | $F_{l0}$ Candidate number | $F_{l1}$ Candidate number |
|-----------|------------|---------------------------|---------------------------|
| C5315     | 2871       | 1824                      | 1826                      |
| C7552     | 4106       | 2932                      | 2933                      |
| S9234     | 6263       | 4583                      | 4585                      |
| S13207    | 9056       | 6669                      | 6670                      |
| S15850    | 10765      | 8601                      | 8599                      |
| S35932    | 18208      | 12454                     | 12453                     |
| S38417    | 24227      | 19374                     | 19375                     |
| S38584    | 21100      | 17885                     | 17887                     |

![Figure 10](image_url)  
**Figure 10** SPS candidate number for different noise percentages (50\% to 0\%)

| Table 5 | SPS analysis results for $N_z = 50\%$ and $n = 16$ |
|--------|--------------------------------------------------|
| Benchmark | Wire count | $F_{l0}$ Candidate number | $F_{l1}$ Candidate number |
| C5315     | 2603       | 1793                      | 1794                      |
| C7552     | 3928       | 2921                      | 2922                      |
| S9234     | 6064       | 4738                      | 4739                      |
| S13207    | 8860       | 6656                      | 6661                      |
| S15850    | 10592      | 8592                      | 8597                      |
| S35932    | 18037      | 12443                     | 12444                     |
| S38417    | 24052      | 19986                     | 19988                     |
| S38584    | 20926      | 17976                     | 17977                     |

| Table 6 | SPS analysis results for $N_z = 50\%$ and $n = 32$ |
|--------|--------------------------------------------------|
| Benchmark | Wire count | $F_{l0}$ Candidate number | $F_{l1}$ Candidate number |
| C5315     | 2603       | 1793                      | 1794                      |
| C7552     | 3928       | 2921                      | 2922                      |
| S9234     | 6064       | 4738                      | 4739                      |
| S13207    | 8860       | 6656                      | 6661                      |
| S15850    | 10592      | 8592                      | 8597                      |
| S35932    | 18037      | 12443                     | 12444                     |
| S38417    | 24052      | 19986                     | 19988                     |
| S38584    | 20926      | 17976                     | 17977                     |
attack accomplishment by removal of this wire, we have deployed multiple Monte Carlo simulations on these circuits to see, whether removal of this high probability skew wire leads to the main circuit or it leads to a fully corrupted circuit.

These Monte Carlo simulations are executed on similar KI patterns in both blocks, while the circuit if unlocked corresponds to a $Diff = 0$ for all KI patterns. Results stated in Table 7, clearly state that removing this wire does not retrieve the main circuit's functionality. Moreover, in the best-case scenario, $Diff = 1$ for all the input patterns is retrieved without any correct keys, and in the worst-case scenario leads to about 37.5% corruption in input patterns for equal KI patterns. Despite former research, this locking technique protects all input patterns as the locking block cannot be removed in SPS attack, and if so, a fully corrupted block remains with no correct key and not a static wrong input. Henceforth, this locking technique supersedes former techniques like TT-Lock [14] and SFLL-hd [24].

5.3 | AGR attack results

We assume the worst-case scenario, in which AppSAT attack terminates with finding all values of traditional key bits. Therefore, only noise-based block is remaining. This attack is simulated in Python programming language, and noise-based and Anti_SAT methods are coupled with benchmarks from ISCAS 85, and ISCAS 87. After the execution of AGR attack, the found wire is compared with locks outputs and if they are identical AGR attack is considered successful. The corresponding results are depicted in Table 8. These results confirm our locking blocks’ superiority to hinder this attack, as the convergence of the KIs is met in the main circuit, and the removal of this wire will end in a faulty circuit. Furthermore, the points which are next to the results of the attack are not mentioned in SPS result list to have a high probability skew, and they have no potential for being detected as lock’s outputs.

5.4 | Area, power and delay

In this section, we discuss the overhead of locking blocks from different aspects.

Locking blocks are not identical in this scheme, therefore, reported values here are not guaranteed to be the most optimal. We have designed locking blocks varying in KIs from 8 to 128 with 50% noise, which add the highest overhead values. Then on, these locking blocks are synthesised and implemented under two different scenarios using the Synopsys Design Compiler with the 45-nm NCSU-FreePDK library [33].

5.4.1 | First scenario

The results of these simulations are reported in Figure 11 in blue, and also SAT iterations needed for each lock are plotted in this logarithmic chart in red. This graph clearly.ravels exponential growth of SAT iterations, where a linear increase in area consumption is reported. This linear growth is estimated to be about 5.17μm² per key.

| n  | 8     | 10    | 12    | 14    | 16    |
|----|-------|-------|-------|-------|-------|
|    | Original lock |  | | | |
|    | 255 | 1023  | 4095  | 16,384 | 65,535 |
|    | Full removal |  | | | |
|    | 24,576 | 393,216 | 6,291,456 | 100,663,296 | 1,610,612,736 |
|    | Logic 1 insertion |  | | | |
|    | 8,320 | 131,584 | 2,099,200 | 16,384 | 536,903,680 |
|    | Logic 0 insertion |  | | | |
|    | 256 | 1024  | 4096  | 33,562,624 | 65,536 |

| Benchmark | Attack result on our method | Attack result on Anti_SAT |
|-----------|-----------------------------|---------------------------|
| C5315     | Fail                        | Succeed                   |
| C7552     | Fail                        | Succeed                   |
| S9234     | Fail                        | Succeed                   |
| S13207    | Fail                        | Succeed                   |
| S15850    | Fail                        | Succeed                   |
| S35932    | Fail                        | Succeed                   |
| S38417    | Fail                        | Succeed                   |
| S38584    | Fail                        | Succeed                   |

FIGURE 11 Design area consumption and SAT iterations are plotted for $n = 8, 10, 12, 14, 16, 32, 64, 128$
5.4.2 | Second scenario

We have coupled these locking blocks (for $n = 64$) with a wire that has an observability lower than 30% of that in the main circuit [14]. These circuits are afterwards compiled, and overhead reports are demonstrated in Table 9. This table states the linear growth in area overhead. Our area and power overhead’s average for $n = 64$ are consecutively 2.74%, 0.95% and our delay overhead is not reported, while our chosen wires in these circuits were not on the circuit’s critical path and not affecting the overall delay. At the same time, these overheads for TT-Lock scheme are 3% and 4.1% in 64-bit locking blocks, which demonstrate our considerable improvement in the case of power overhead, and our better result in area of overhead. These improvements are due to utilising the least number of XOR gates and omitting the resynthesis procedure.

6 | CONCLUSION

We present the noise-based locking method to mitigate the removal attacks. In this method, we benefit from two coupled locking blocks that together act as a low corruption block, while having an adjustable probability skew from 50% in the output. Overall, our method thwarts removal attacks without adding extra dummy parts, and in a relatively lower amount of overhead in comparison to other methods. Additionally, our method has a higher SAT resilience than the Anti_SAT block (with 17% improvement for $n = 14$). Moreover, TT-Lock method, which has a lower security level and protects finite input patterns after removal attacks, is compared with our method which shows our 0.26% and 3% improvement, respectively, in the area and power overhead for $n = 64$.

REFERENCES

1. Innovation is at risk losses of up to $4 billion annually due to IP infringement (2008)
2. Quadir, S.E., et al.: A survey on chip to system reverse engineering. J. Emerg. Technol. Comput. Syst. 13(1), 61–634 (2016)
3. Pecht, M., Bogus, S.T.: Electronic manufacturing and consumers confront a rising tide of counterfeit electronics. IEEE Spectr. 43(5), 37–46 (2006)
4. Rostami, M., Kooshanfar, F., Karri, R.: A primer on hardware security: models, methods, and metrics. Proc. IEEE. 102(8), 1283–1295 (2014)
5. Agarwal, N., Amit, K., Singh, P.K.: Survey of robust and imperceptible watermarking. Multimed. Tools Appl. 78(7), 8603–8633 (2019)
6. Patnaik S., A Modern Approach to IP Protection and Trojan Prevention: Split Manufacturing for 3D ICs and Obfuscation of Vertical Interconnects. IEEE Transactions on Emerging Topics in Computing, 1–1 (2019). https://doi.org/10.1109/tetc.2019.2933572
7. Cocchi, R.P., et al.: Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing. US Patent. 8(510), 700 (August 13 2013)
8. Roy, J.A., Kooshanfar, F., Markov, I.L.: Epic: Ending piracy of integrated circuits. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1069–1074.ACM (2008)
9. Dupuis, S., Flottes, M.-L.: Logic locking: a survey of proposed methods and evaluation metrics. J. Electron. Test. 1–19 (2019)
10. Rajendran, J., et al.: Security analysis of logic obfuscation. In: Proceedings of the 49th Annual Design Automation Conference, pp. 83–89.ACM (2012)
11. Xie, Y., Srivastava, A.: Anti-sat: Mitigating sat attack on logic locking. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 38(2), 199–207 (2018)
12. Yasin, M., et al.: Sarlock: Sat attack resistant logic locking. In: 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 236–241. IEEE (2016)
13. Yasin, M., et al.: On improving the security of logic locking. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 35(9), 1411–1424 (2016)
14. Yasin, M., et al.: What to lock?: Functional and parametric locking. In: Proceedings of the on Great Lakes Symposium on VLSI 2017, pp. 351–356. ACM (2017)
15. Alasad, Q., Yuan, J.: Logic obfuscation against IC reverse engineering attacks using PLGs. In: 2017 IEEE International Conference on Computer Design (ICCD), pp. 341–344. IEEE (2017)
16. Subramaniam, P., Ray, S., Malik, S.: Evaluating the security of logic encryption algorithms. In: 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 137–143. IEEE (2015)
17. Yasin, M., et al.: Removal attacks on logic locking and camouflaging techniques. IEEE Trans. Emerg. Topics Comput. (2017)
18. Xu, X., et al.: Novel bypass attack and BDD-based trakoff analysis against all known logic locking attacks. In: International Conference on Cryptographic Hardware and Embedded Systems, pp. 189–210. Springer (2017)
19. Khaleghi, S., Zhao, K.D., Rao, W.: IC piracy prevention via design withholding and entanglement. In: The 20th Asia and South Pacific Design Automation Conference, pp. 821–826. IEEE (2015)
20. Rajendran, J., et al.: Fault analysis-based logic encryption. IEEE Trans. Comput. 64(2), 410–424 (2013)
21. Yasin, M., et al.: Security analysis of anti-SAT. In: 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 342–347. IEEE (2017)
22. Meng, L., et al.: Provably secure camouflaging strategy for IC protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. (2017)
23. Yasin, M., et al.: Camoper turb: Secure IC camouflaging for minterm protection. In: 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–8. IEEE (2016)
24. Yasin, M., et al.: Provably-secure logic locking: From theory to practice. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1601–1618 (2017)

| Benchmark | Method | Area overhead (%) | Power overhead (%) | Delay overhead (%) |
|-----------|--------|-------------------|--------------------|-------------------|
| S35932    | TT-Lock| 4                 | 6                  | 1.4               |
| S35932    | Our method | 2.71            | 1.5                | Unaltered         |
| S38417    | TT-Lock| 2.9               | 3.1                | 0.1               |
| S38417    | Our method | 2.8             | 0.32               | Unaltered         |
| S38584    | TT-Lock| 2.1               | 3.2                | 0.5               |
| S38584    | Our method | 2.72            | 1.03               | unaltered         |

**Table 9** Comparison metrics (area, power and delay) of TT-Lock and noise-based method versus original circuit for $n = 64$
25. Kaveh, S., et al.: On the approximation resiliency of logic locking and IC camouflaging schemes. IEEE Trans. Inform. Forensics Secur. 14(2), 347–359 (2018)

26. Yang, F., Tang, M., Sinanoglu, Oz.: Stripped functionality logic locking with hamming distance-based restore unit (sll-hd)-unlocked. IEEE Trans. Inf. Forensics Secur. 14(10), 2778–2786 (2019)

27. Shakya, B., et al.: CAS-lock: a security-corruptibility trade-off resilient logic locking scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. 175–202 (2020)

28. Sengupta, A., Sinanoglu, O.: CAS-unlock: Unlocking CAS-Lock Without Access to a Reverse-Engineered Netlist.

29. Xie, Y., Srivastava, A.: Mitigating SAT attack on logic locking. In: International Conference on Cryptographic Hardware and Embedded Systems, pp. 127–146. Springer (2016)

30. Hansen, M.C., Yalcin, H., Hayes, J.P: Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. IEEE Des. Test Comput. 16(3), 72–80 (1999)

31. Biere, A.: Lingeling, plingeling and treengeling entering the sat competition 2013. Proc. SAT Compet. 1, (2013)

32. Brglez, F., Bryan, D., Kozminski, K.: Combinational profiles of sequential benchmark circuits. IEEE Int. Symp. Circuits Syst. 3, 1929–1934 (1989)

33. James, E.S., et al.: Free PDK: An open-source variation-aware design kit. In: 2007 IEEE International Conference on Microelectronic Systems Education (MSE’07), pp. 173–174. IEEE (2007)

---

**How to cite this article:** Rezaei A, Mahani A. Noise-based logic locking scheme against signal probability skew analysis. *IET Comput. Digit. Tech.* 2021;15:279–295. [https://doi.org/10.1049/cdt2.12022](https://doi.org/10.1049/cdt2.12022)