IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 October 2023, accepted 9 November 2023, date of publication 13 November 2023,
date of current version 20 November 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3332543*

RESEARCH ARTICLE

# Hybrid Shielding: Amplifying the Power of Camouflaging and Logic Encryption

**NIKHIL SAXENA AND RANGA VEMURI, (Senior Member, IEEE)**
Electrical and Computer Engineering Department, University of Cincinnati, Cincinnati, OH 45220, USA
Corresponding author: Nikhil Saxena (saxenanl@mail.uc.edu)

**ABSTRACT** In the semiconductor industry, protecting Integrated Circuits (IC) throughout the IC supply chain has become a major concern. In-depth research has been done on logic encryption, split manufacturing, and layout camouflaging to safeguard ICs against attacks at various stages of the supply chain. In this work, we introduce a hybrid, method called Hybrid Shielding (which amplifies the power of camouflaging and logic locking) to protect ICs at each stage of the supply chain, including the foundry, the testing facility, and the end user. We take advantage of the spin-based device, called the Giant Spin-Hall Effect (GSHE) switch, multi-functionality, post-fabrication reconfigurability, and run-time polymorphism to make dynamic camouflaging resistant to SAT-based attacks and test-data mining-based attacks. These characteristics are not available to designers in CMOS. We define two metrics for circuit nodes: stability and weight. Hybrid Shielding replaces all of the selected gates with polymorphic gates. It uses a simulator to ascertain the internal state of the selected nodes. The camouflaged internal state will be used to corrupt the functionality of the primary outputs. The resulting locked circuit has high output corruption rates and is resilient to the SAT attack, Hack Test, as well as several other attacks. These results are demonstrated experimentally using standard benchmark circuits.

**INDEX TERMS** Camouflaging, emerging devices, hardware security, hack tests, logic encryption, output corruption rate, polymorphic gates, SAT attack.

## I. INTRODUCTION

With the globalization of design, the increasing cost of fabrication, and the complex manufacturing process of ICs, hardware security has become a prominent issue. The increasing cost of IC fabrication has forced the use of potentially untrusted offshore foundries to fabricate ICs. At every step of the IC supply chain, ICs have become vulnerable to several attacks including reverse engineering, overproduction, counterfeiting, trojan insertion, and IP theft [1].[1]

A typical IC supply chain is depicted in Figure 1 along with potential attacks at each stage. The team at the design house will design SoC once they have the system specifications. However, the design is not secure because reverse engineering and IC piracy are still possible at this point. The gate-level netlist is used to generate a layout, which the offshore foundry fabricates. Yet again, the procedure is not secure and may be attacked by one of these intruders: reverse engineering, IC theft, and hardware trojans. Additionally, offshore foundries have serious concerns about overbuilding IC. Following the development of the wafer, testing, and packaging are crucial next steps that can be attacked by hardware trojans, reverse engineering, and counterfeiting. The final product can then be obtained to deliver it to the user, and this is also vulnerable to attack from hardware trojans, counterfeiting, and reverse engineering [1].

Design-for-trust schemes have emerged in recent years to overcome these threats. Among the most promising schemes are logic encryption [1] and layout camouflaging [2] which

The associate editor coordinating the review of this manuscript and approving it for publication was Dušan Grujić.

[1]Certain sections and methodologies within this paper were previously featured in a conference paper referenced as "Nikhil Saxena, Ranga Vemuri, "Hybrid Shielding: Amplifying the Power of Camouflaging and Logic Encryption", 66th IEEE international midwest symposium on circuits and systems, 2023". Nonetheless, this represents an expanded iteration of the paper, incorporating novel methodologies.

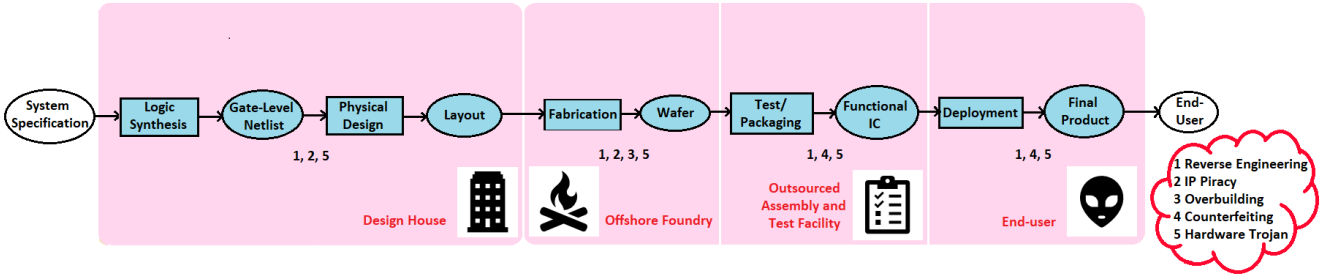



**FIGURE 1.** IC supply chain.

can thwart potential attacks at multiple stages of the supply chain.

### A. SUMMARY OF IP PROTECTION METHODS

#### 1) LOGIC ENCRYPTION

In this method, designers add key-controlled locking gates to an integrated circuit (IC) to corrupt the circuit's functionality unless the right key vector is used. The simplest locking technique involves using XOR/XNOR/AND/OR/MUX gates to insert additional locking logic [3]. The locked IC is activated at the user end by the trusted design house. Random encryption (EPIC) [3], AntiSAT [4], stripped functionality logic locking (SFLL) [5], SARLock [6], Strong AntiSAT [7], and SRTLock [8] are just a few of the encryption methods that have been introduced. A significant Output Corruption Rate (OCR) in the encrypted circuit when a wrong key input is used is essential [9]. None of the proposed provide significant OCR on their own, and the OCR is relatively small even with the addition of random key gates [9]. Additionally, the secure implementation of tamper-proof memory (TPM), is crucial to the overall security of logic locking [26]. TPM is an active research topic [26].

#### 2) IC CAMOUFLAGING

Reverse engineering (RE) attacks can be mitigated through IC camouflaging. In this method, IC's layout is altered such that it becomes nearly impossible to decipher the underlying functionality [10]. It replaces a few selected gates with their camouflaged equivalents [11]. As delaying proceeds from the top, camouflaged gates appear to be identical, but different control inputs make them behave differently. Several camouflaging strategies have been suggested, for both traditional CMOS layouts and emerging devices. Among them are look-alike gates [2], threshold-dependent camouflaging [12], [13], and obfuscated interconnects [14].

#### 3) SPLIT MANUFACTURING

Split manufacturing refers to the division of a design into two segments that correspond to the BEOL and FEOL metal layers. These parts are manufactured in different foundries and then stacked together [27]. Split manufacturing is another layout-level defense mechanism. It can protect against RE by an untrusted foundry, but not by an end-user [11].

To conclude, even though IP protection methods have been suggested to protect the supply chain from malicious entities, each of these solutions has some drawbacks. Although logic locking has the potential to safeguard the entire IC supply chain, in its current form, resilience comes at the expense of a very low OCR. A TPM that stores the secret key is also necessary to protect the key.

### B. EMERGING DEVICES IN HARDWARE SECURITY

In comparison to their CMOS counterparts, emerging devices, such as nanowire transistors and carbon or spin-based devices may offer lower power dissipation and higher integration densities [28]. A polymorphic gate can implement various Boolean functions at runtime, with the functionality being determined by an internal/external control mechanism. From a security perspective, polymorphism is the most promising feature of many emerging devices. Unlike other emerging devices, spin devices, such as GSHE devices [10], have inherent run-time polymorphism and post-fabrication reconfigurability capabilities. Polymorphic logic has been used in recent works [9], [10], [29] for logic locking to create static camouflaging. However, dynamic camouflaging, which has not yet been fully explored, promises to exploit the full potential of polymorphic devices. Exploring dynamic camouflaging along with logic locking is the main objective of this paper.

### C. DYNAMIC CAMOUFLAGING

Dynamic IC camouflaging aims to counteract all of the potential attacks shown in Figure 1. Dynamic camouflaging involves obfuscating circuit-level functionality during runtime. We investigate dynamic camouflaging using a polymorphic GSHE device in conjunction with logic encryption. We identify security and OCR as two intertwined design variables, particularly to thwart error-tolerant attacks like machine learning attacks. We focus on these scenarios because we think they are significant.

However, the number of logic gates that will need to be replaced by polymorphic gates is a concern. A significant number of gate replacements could result in an undesirable performance overhead. Choosing the gates that should be replaced requires careful analysis. OCR could be low if

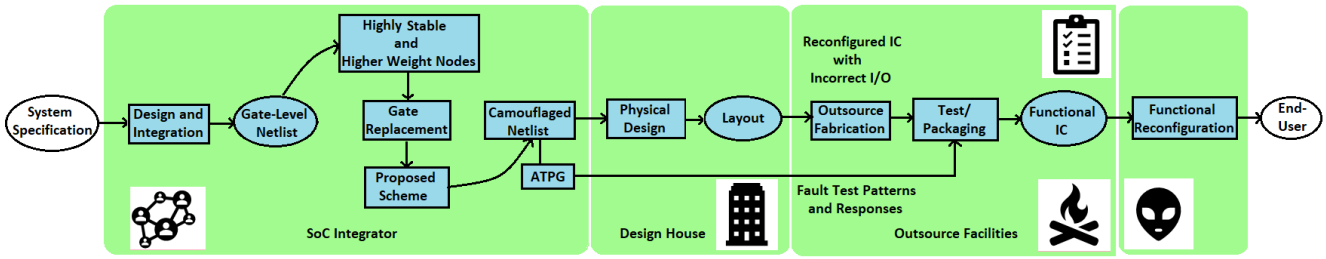



**FIGURE 2.** Inbuilt hybrid shielding in IC supply chain.

**TABLE 1.** IC protection techniques versus untrusted entities in IC supply chain.

| Technique | Design House | Offshore Foundry | Test Facility | End User | OCR | PPA |
|---|---|---|---|---|---|---|
| Logic Encryption | √ | √ | √ | √ | Low | Low |
| Logic Camouflaging | × | × | × | √ | Moderate | Moderate |
| Split Manufacturing | √ | × | × | × | NA | NA |
| Dynamic Camouflaging | √ | √ | √ | √ | High | Moderate |
| Hybrid Shielding | √ | √ | √ | √ | High | Low |

the designer chooses an approach that offers more SAT resilience [9].

### D. HYBRID SHIELDING

We suggest a hybrid encryption technique that makes use of both logic encryption and dynamic IC camouflaging to overcome the aforementioned problems. High OCR and SAT resilience can both be preserved using hybrid shielding along with protection at all stages of the IC supply chain. Additionally, this scheme is area and power-efficient due to the smaller number of gates that must be replaced with polymorphic ones. As we demonstrate in this work, hybrid shielding is effective against both exact [15], [16] and approximate SAT (AppSAT) [21] attacks. We explain how the supply chain can be protected from end-to-end using hybrid shielding while avoiding the security risks at unreliable foundries, test facilities, and end-users (Figure 2). In the illustration in Table 1, the cross sign stands for not protecting at that level. However, protection for that level is provided by a right sign.

Figure 2 shows how hybrid shielding makes the supply chain secure at each stage. Key features of the proposed scheme are as follows:

1) We use a GSHE device-based polymorphic gate to propose a security primitive. We carefully select certain stable internal signals along with high weight from the cone of influence of the primary output and then replace the selected signals from polymorphic gates. We also determine the internal state of the selected signals using a simulator.
2) We use dynamic camouflaging and share the key inputs among the polymorphic gates to reduce the area and power consumption.
3) We use this internal state along with a *SAT-resilient block* to strip some functionality of primary outputs. To correct the functionality of primary outputs, we use a *reinstate block*. This step is specially included to thwart the SAT attack and other related attacks.
4) To use key inputs effectively, we split the keys into three subsets. One set is used as a control input for polymorphic gates. The second set is used in the SAT-resilient block and the third set (which might even be a single key bit) is used to reinstate the output.
5) The proposed encryption scheme can provide a near 31% average output corruption rate without augmenting the SAT-resilient scheme with a random encryption scheme.
6) Our method is resilient to Hack-Test [34], AppSAT [21], SAT-attack [15] and sensitivity attack [23].
7) We also present a standard method for calculating hybrid shielding Power-Performance-Area (PPA) overhead without the use of emerging device model files.

This paper is organized as follows: Section II describes the background and motivation behind the work. Section III describes the proposed logic obfuscation scheme, Hybrid Shielding. Section IV describes the protection provided by the proposed method against existing attacks and the experimental results. Section V includes a case study to show why stable signals are important. Section VI contains PPA overhead calculation. Section VII contains concluding remarks.

## II. BACKGROUND AND MOTIVATION

This section will go over the significant increase in logic locking as well as the associated attacks. The adaptation of logic locking to static camouflaging will also be discussed. We will also discuss preliminary research on dynamic camouflaging.

### A. LOGIC LOCKING, STATIC LAYOUT CAMOUFLAGING AND SAT-BASED ATTACKS

When it comes to defeating SAT, logic locking has a limitation in providing low OCR. The preliminary research

focused on two primary research gaps in camouflaging. The first is the selection of camouflaged nodes, and the second is the design of camouflaged cells. Because of their high layout cost (measured in PPA), the majority of camouflaging schemes have low practical implementation. A camouflaged gate with three boolean functions (XOR+NAND+NOR) uses 5.5X more power, 4X more area, and 1.6X more delay overhead than a CMOS-based NAND gate. A CMOS-based NOR gate and XOR gate, on the other hand, use 5.1X and 0.8X power, 4X and 1.2X area, and 1.1X and 0X delay overhead, respectively. As a result, the type of gate replacement influences power, area, and delay overhead. When 25% of the gates are camouflaged, the overhead for threshold voltage-defined (TVD) logic gates is 4%, 41%, and 44% for power, area, and delay, respectively. However, the overhead for 100 percent camouflage is 14%, 150%, and 82% for power, area, and delay, respectively. Because of the high overhead, current logic camouflaging schemes can only hide a limited number of gates, compromising security.

Analytical security threats aimed at camouflaged (or locked) ICs were proposed in [15] and [16]. Using Boolean satisfiability (SAT) formulations, these attacks infer the original circuit functionality. SAT is a powerful de-obfuscation technique that uses an Oracle IC and a Boolean satisfiability engine to quickly eliminate incorrect key inputs from an obfuscated circuit. It arrives at the class of correct key values after removing the classes of keys that result in incorrect outputs. There are numerous SAT-resilient schemes in the literature, including [5] and [6]. However, the majority of these methods were found to be vulnerable to the Boolean sensitivity attack [23]. Each of these schemes used either point function encryption or stripped functionality to secure the design. As a result, as illustrated in figure 3, these methods have low output corruptibility [9]. FLL [30], an encryption technique, improved output corruption as well as SAT resiliency. To find a roughly accurate key, Shamsi et al. [21] re-model the SAT attack method into an approximation method. It breaks down such compound schemes into their low-corruptibility component by ''peeling off'' the high-corruptibility portion. This is accomplished by relaxing SAT's exactness constraint and instead allowing a key that meets a specified error threshold. The resulting key is a version of the original design that produces the correct output for the vast majority of input patterns, except for a few within the error threshold. This method is useful when using an SAT-hardened encryption scheme, but an approximation of the original design will suffice in most cases.

### B. GSHE DEVICE

In recent years, several algorithms have been proposed to select logic gates in a circuit and replace them with polymorphic gates. Patnaik et al. [10] proposed using giant spin-Hall effect (GSHE) switch-based polymorphic gates that perform all 16 2-input Boolean functions and suggested exchanging at least 5% of nodes in larger circuits to achieve SAT-resiliency and replacing gates by 40% to 100% in smaller circuits. Datta et al. [17] first proposed the GSHE device. The switching element in the spin domain is chosen as the GSHE device, which operates in a two-magnet system using the dipolar coupling phenomenon [31]. The GSHE device also has excellent CMOS process compatibility, according to [18].

The central component of the proposed primitive, the GSHE switch, is depicted in Figure 4(a). The nanomagnets (NM) for write (W; green, bottom) and read (R; green, top) are located above the heavy metal spin Hall layer (purple, bottom). The nanomagnets W-NM and R-NM exhibit negative mutual dipolar coupling. On top of the R-NM are two fixed ferromagnetic layers (green) with anti-parallel magnetization directions. As shown by the large black arrow in Figure 4(a), a charge current applied to the bottom layer causes spin accumulation of one polarity (green spin-up spheres) in the transverse direction (yellow arrows). This spin-polarized current [32] then provides a spin transfer torque (STT) to the W-NM. The STT shifts the W-NM from one stable state to another, causing the R-NM to shift in the opposite direction. Now, one of the fixed ferromagnets on top will be parallel to the direction of R-magnetization NM, while the other will be anti-parallel. The parallel path has a lower resistance for a charge current traveling from/to the corresponding top contact to/from the output terminal. To begin the read-out phase, voltages are applied to the top contacts (+V and −V). Depending on the polarity of the voltage applied to the low-resistance path, the output current flows inward or outward. This is the binary result of the GSHE switch operation (4(b)).

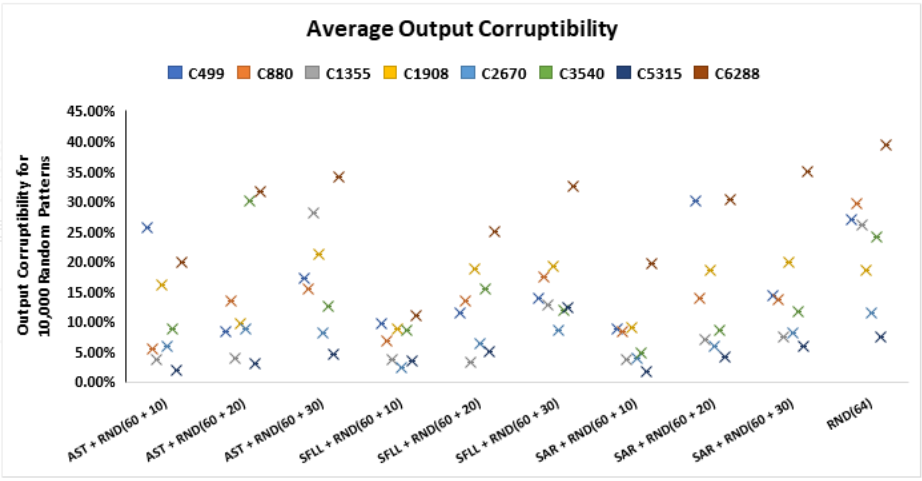
### C. STATIC TO DYNAMIC CAMOUFLAGING TOWARDS HYBRID SHIELDING

An emerging research problem is the efficient use of polymorphic gates for hardware security. The effective integration of dynamic camouflaging and logic encryption is still unexplored. In this paper, we propose a novel method for enhancing the effectiveness of dynamic camouflaging and logic encryption. Figure 5 depicts the overall structure of the proposed scheme. We combined both methods, maximizing their benefits, and proposed a novel and distinct technique. We call it hybrid shielding.

Figure 6 depicts the internal steps required to achieve hybrid shielding on a c17 benchmark from the ISCAS-85 [19]. The selected high stability and high weight internal signals are shown in step I of figure 6. In step 2, the selected signals are replaced by dynamic polymorphic gates, and in step 3, the proposed logic encryption technique is integrated, making it hybrid. In the following section, we'll review figure 6 in greater depth.

## III. PROPOSED LOGIC ENCRYPTION METHOD

As previously stated, the output corruption rate for existing SAT-resilient schemes is far too low. Although the dynamic behavior of polymorphic gates can be used to increase the




**Average Output Corruptibility:** AST+RND(60+10)=10.87%, AST+RND(60+20)=13.61%, AST+RND(60+30)=17.55%, SFLL+RND(60+10)=6.79%, SFLL+RND(60+20)=12.30%, SFLL+RND(60+30)=16.08%, SAR+RND(60+10)=7.45%, SAR+RND(60+20)=14.74%, SAR+RND(60+30)=14.43%, RND(64)=22.93%

**FIGURE 3.** Average Output Corruptibility for AntiSAT (AST), SFLL, SARLock (SAR), and Random (RND) Encryption Schemes. Average Output Corruptibility for AST+RND(60+10)=10.97% (here 60 is an allotted key size for AST and 10 is an allotted key size for RND), similarly for AST+RND(60+20)=13.61%, AST+RND(60+30)=17.55%, SFLL+RND(60+10)=6.79%, SFLL+RND(60+20)=12.30%, SFLL+RND(60+30)=16.08%, SAR+RND(60+10)=7.45%, SAR+RND(60+20)=14.74%, SAR+RND(60+30)=14.43% and for only random encryption using 64 key sizes is RND(64)=22.93%.

**TABLE 2.** Comparison of OCR for most sensitive and most stable signals.

| Sl No | Benchmark | OCR for Sensitive Nodes (%) | OCR for Stable Nodes (%) |
|---|---|---|---|
| 1 | C432 | 40.76 | 31.92 |
| 2 | C499 | 4.66 | 49.61 |
| 3 | C880 | 21.05 | 38.13 |
| 4 | C1355 | 25.78 | 30.20 |
| 5 | C1908 | 11.74 | 28.37 |
| 6 | C5315 | 15.65 | 19.71 |
| 7 | C6288 | 40.29 | 34.89 |
| 8 | C7552 | 13.99 | 15.62 |
| 9 | **Total %** | **21.79%** | **31.06%** |

corruption rate, the current method requires replacing many gates [10]. We propose a method for replacing a few selected gates in a circuit with polymorphic gates in an efficient manner.

### A. SELECTION OF INTERNAL SIGNALS

Careful selection of internal signals can improve output corruptibility while preserving the SAT-resiliency of encrypted circuits. To select the internal signals, we use the input cone-of-influence of the output signals. The algorithm 1 classifies internal signals based on their stability and weight. If the node in the cone of influence is highly stable, it is preferred to camouflage. In comparison to the most sensitive nodes, camouflaging the stable nodes increases the sensitivity[2] of primary outputs. We used camouflage to transform the most stable nodes into the most sensitive nodes. Most sensitive nodes in the circuit, on the other hand, are already sensitive and do not change much after camouflaging. A comparison of the two can be found in table 2. All of the results are run for 60 key inputs and 45 camouflaged gates.

For ease of implementation, we set the level $l$ value to 4. If the number of nodes chosen for stability calculation $N$ is less than the number of polymorphic gates $n$, the designer can increase the $l$ counts. Lines 4 to 6 of the algorithm 1 show a group of fan-in nodes from the input cone of influence of primary outputs. When selecting the signals, keep in mind that the primary inputs, reset, and clock signals are ignored. For small circuits, we wanted $N$ to be nearly double $n$, and for larger circuits, we wanted any count higher than double $n$. This provides enough room to calculate and compare stability and yields better results in terms of output corruption. Line 7 invokes the simulator and retrieves values for $N$ signals $m$ times. Lines 8 to 21 represent the stability calculation for each signal in $N$. Signals in $N$ may be connected to multiple primary outputs; we record this for each selected signal and define its weight. If any signal demonstrates high stability with a larger weight, the output corruption rate is excellent. Weight calculation is represented by lines 22 to 24 of the algorithm 1. Lines 26 to 42 show the signal selection process. Though it is ultimately up to the designer, we made every effort to provide the best possible combination of extremely stable and heavily weighted signals.

### B. DISTRIBUTION OF KEYS

The total number of polymorphic gates represented by $n$ determines the distribution of keys. User-defined parameters include key input count ($k$), the number of configurations that each polymorphic gate can perform ($l$), and $n$. The reinstate block is controlled by the $s_1$ set of key inputs,

[2]**Sensitivity of Outputs**:- Sensitivity of output refers to the primary output values changing along with the input vectors when the camouflaged gates are given incorrect key input values. Any output that fluctuates often is more sensitive to input vectors with incorrect key inputs delivered to the camouflaged gates.

**Algorithm 1** *Selection of Signals*

```
procedure SELECT_SIGNALS(Circuit C, # of polymorphic signals n, # of patterns m, level l)
    N, output_vectors, stability_of_signals ← {}
    weight_of_signals, selected_ nodes ← {}
    for i = 0; i <= l; i++ do
        N ← N + signals in fanin cone of the primary outputs of C which are 1 levels away from the primary outputs of C
    end for
    output_vectors ← Extracted values for N signals for m times using simulator
    for index from 0 to N-1 do
        for v in output_vectors do
            stability = 0
            for count in range(0, m-1) do
                value1 = output_vectors[count]
                value2 = output_vectors[count + 1]
                if value1[index] == value2[index] then
                    stability += 1
                end if
            end for
            % stability = stability/m
            stability_of_signals[v] = [% stability]
        end for
    end for
    for s in stability_of_signals do
        weight_of_signals[s] ← weight of s w.r.t. primary outputs
    end for
    sort stability_of_signals in descending order
    for signal in stability_of_signals do
        if weight_of_signals[signal] > 50% of Primary Outputs in C and stability_of_signals[signal] > 60% then
            add signal in Selected_ nodes
        end if
        if len(selected_nodes) == n then
            exit
        end if
    end for
    if len(selected_nodes) != n then
        reduce weight_of_signals condition till 2% and repeat step in line 26 to 33
    end if
    if len(selected_nodes) != n then
        reduce stability_of_signals condition by 10% and repeat step in line 26 and 33
    end if
    if len(selected_nodes) != n then
        add signal based on stability of signals starts considering from larger to smaller
    end if
    return selected_ nodes
end procedure
```

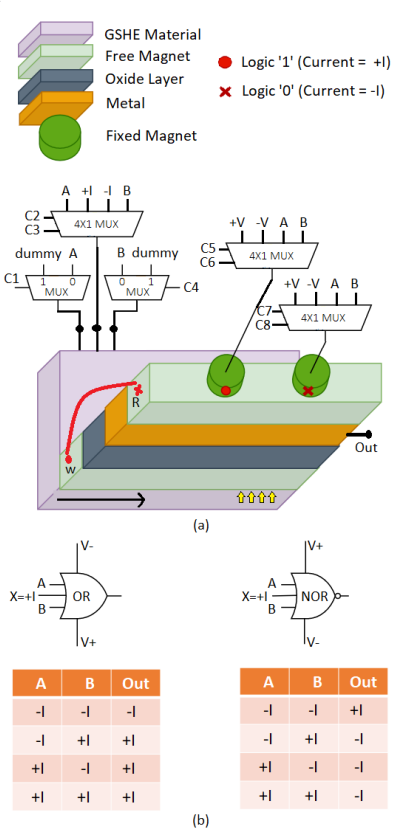



**FIGURE 4.** Dynamic behavior of GSHE switch-based polymorphic device.

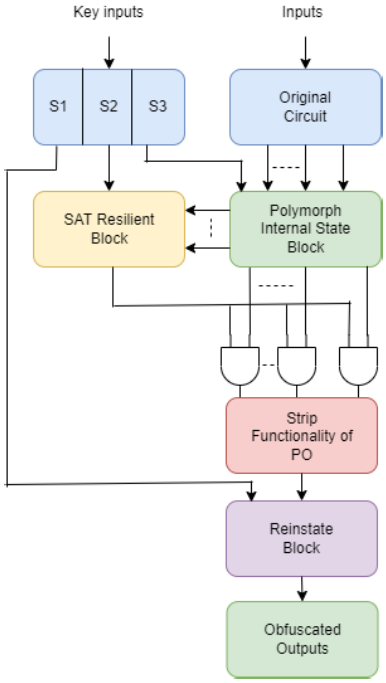



**FIGURE 5.** Proposed encryption scheme block diagram.

the SAT-resilient block by the $s_2$ set of key inputs, and the internal state block by the $s_3$ set of key inputs. As shown in step 3 of fig 6, it$|s_2|$ is always equal to $n$. $n$ is 2 in this

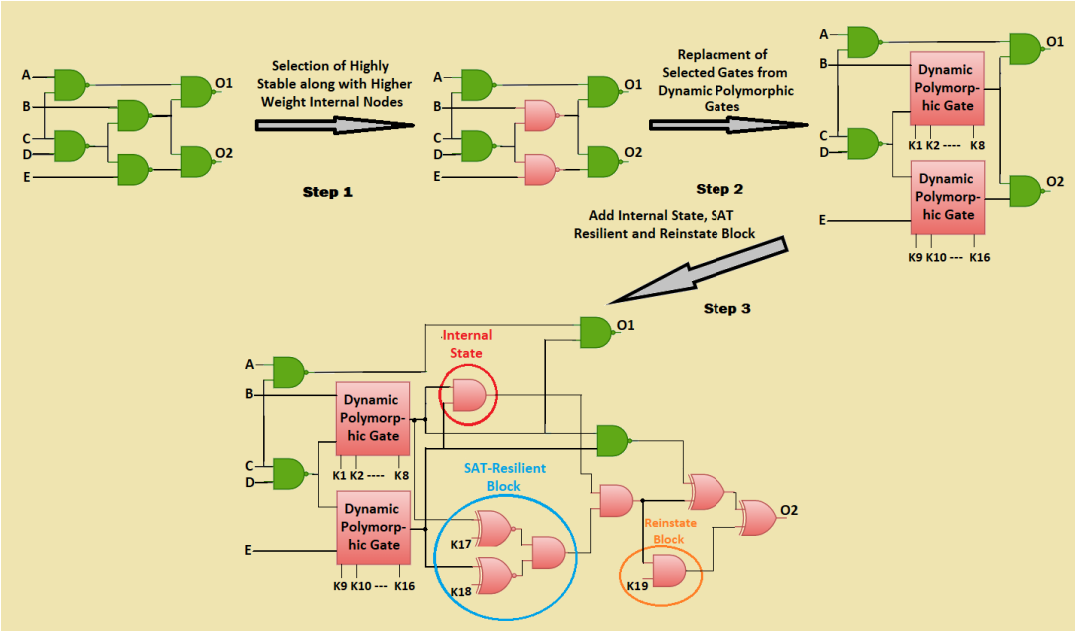



**FIGURE 6.** Hybrid shielding step by step.

case. In the figure 6, the key inputs $K17$ and $K18$ represent the set $|s_2|$, which is exactly equal to $n$. The number of keys for the reinstate block is determined by the type of reinstate block. This is depicted in Fig. 5 as set $|s_1|$. Using the equation $2 * [log_2 l]$, it takes eight key bits to control a GSHE switch-based dynamic polymorphic gate with 16 configurations. However, for CMOS logic gates, a single key bit is all that is required. This can also be seen in step 3 of fig 6, where $K19$ represents the reinstate block, and there is only one because we used a CMOS type gate. As long as there is enough space, the keys can be shared among different polymorphic gates. $|s3|$ is obtained by solving the equation $|s3| \leftarrow k - (|s1| + |s2|)$. In step 3 of Fig. 6, the set of key inputs for $s3$ is represented by $K1$ to $K16$. Although we did not demonstrate key sharing in fig 6, key sharing is included in the actual implementation for the internal state block.

### C. REPLACEMENT OF INTERNAL SIGNALS

After the signal selection is complete, the next step is to replace all selected signals with polymorphic gates. We used polymorphic gates based on GSHE switches to replace the internal signals. The GSHE switch was chosen because it can perform all 16 boolean operations. Previous research has shown that polymorphic gates that can conceal 16 boolean functionalities outperform polymorphic gates that can only perform a limited number of boolean operations. As a result, we avoided comparing various dynamic polymorphic gates with multiple boolean functionalities, such as two, three, and more. Step 2 of Figure 6 depicts the procedure for replacing the signals. Each dynamic polymorphic gate operation is governed by control inputs. Control inputs are key inputs. We used a total of 8 key inputs for each dynamically camouflaged gate. The integration of key inputs by the polymorphic gate is depicted in Figure 7. The $16 \times 1$ MUX's inputs are Dynamic A and B. We used the term "dynamic" because the circuit behaves differently/dynamically when the same primary inputs are used with the incorrect fixed key input multiple times.

### D. POLYMORPH INTERNAL STATE BLOCK

A simulator is used to extract an internal state. The total number of polymorphic gates is divided into four equal parts, giving rise to four internal states. As a result, the simulator needs four input vectors. We created random input vectors and used them as inputs to simulate the original circuit. The internal state is derived from the output of the simulator. Designers can obstruct reverse engineering by dividing internal states into several tiny sections. However, there is a risk that SAT resilience will be lost if internal state lengths are too short. Divide it into no more than four parts for 45 polymorphic gates advised. However, if the number of polymorphic gates is greater, the designer can increase the number of internal states. In the following section, the extracted internal states are combined with the SAT-resilient block, and the primary outputs are locked. Step 3 of Figure 6 depicts a visual representation of an internal state block.

### E. SAT-RESILIENT BLOCK AND REINSTATE BLOCK

The SAT-resilient block is used to provide resistance to SAT and related attacks such as appSAT, sensitization, and partial break. This block contains each dynamic polymorphic gate that has been assigned the appropriate key. Keep in mind that the scheme will never use the key inputs used in this block again. As shown in step 3 of 6, the internal states are ANDed

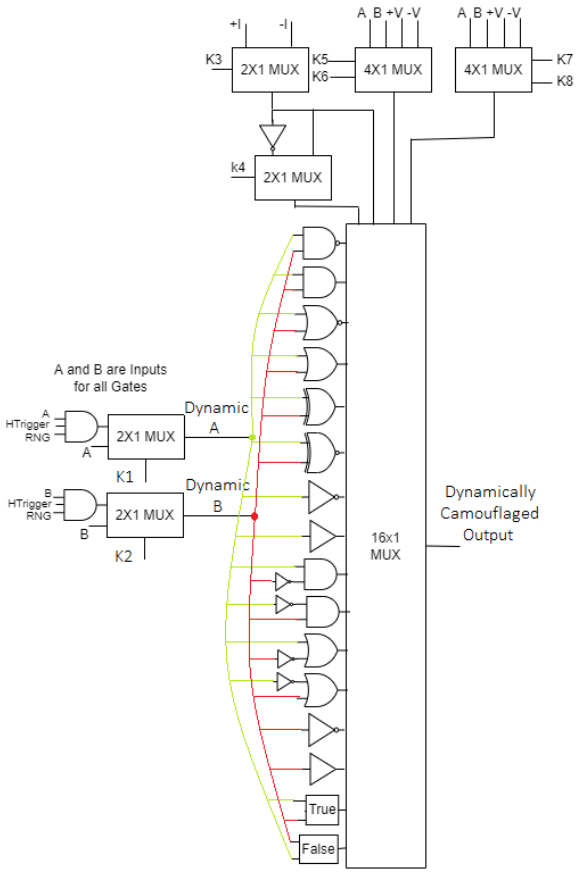



**FIGURE 7.** Dynamic polymorphic gate with 16 functionalities.

with the output of this block. After that, the ANDed output is used to corrupt the primary outputs.

The reinstate block can be used to restore the corrupted primary outputs. We use the key input ANDed with the combined output produced by the SAT resilient block and internal state block to recover the corrupted output, as shown in step 3 of Fig. 6.

## IV. RESULTS AND EXPERIMENTS

**SetUp:** We used a 2 GHz AMD Ryzen 7 2700X eight-core processor with 16 GB of RAM for the experiments. For evaluation, we used the ISCAS-85 [19] benchmarks. We used 60 key inputs in the experiments. We compared two sets of polymorphic gates: 45 polymorphic gates and 30 polymorphic gates. For ease of experimentation, we fixed the type of reinstate block to CMOS, so we only used one key for reinstate block.

### A. PREVENTING FROM UNTRUSTED FOUNDRY

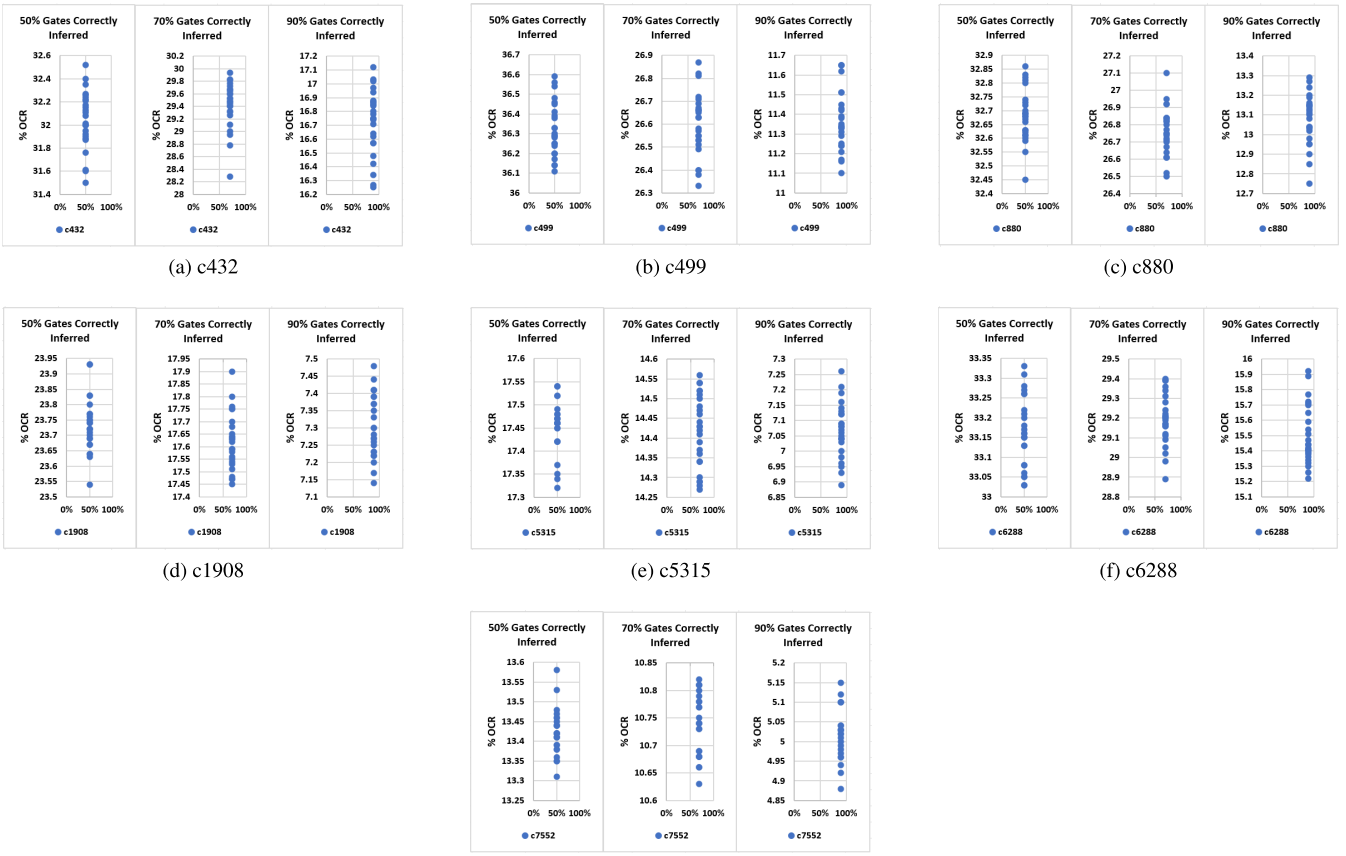
An attacker in the foundry can easily deduce the IP used in CMOS, but the GSHE switch-based gates appear to be white-box devices with no fixed functionality. Because the physical layout of the Boolean gates used in the GSHE switch is optically indistinguishable, optical inspection-guided RE is difficult. Because it is based on post-fabrication reconfigurability, our strategic approach here is resistant to ''inside foundry'' attacks. The ability of GSHE switch gates to be reconfigured after fabrication makes determining their exact functionality more difficult. As shown in Figure 7, it has a solution space of 16 $\times$ 16 possible netlists, with only one of them being correct for a random gate-guessing attack. The logical inputs a and b are fed in parallel with all sixteen possible Boolean functions, and the outputs of those gates are connected to a 16-to-1 MUX with four select lines/key inputs.

The risk model that we use for the security analysis of an untrustworthy foundry is as follows:

- A malicious foundry worker has access to the physical design, which includes the material and layout specifications of the GSHE switch gates and the chip's interconnects. An adversary in a foundry can easily determine the dimensions and material makeup of each GSHE switch-based gate's nanomagnet, and this knowledge allows the adversary to determine the nanomagnet's magnetic characteristics, such as saturation magnetization, energy barrier, and critical ME field for switching. These design details, however, reveal nothing about the gate's intended functionality.
- They possess knowledge of the gate selection algorithm at its core, the quantity and varieties of camouflaged gates, yet they remain unaware of the specific functions offered by each gate.
- Given that the functional chip is presumed to remain inaccessible for security scrutiny in the public market, the individual must resort to ''inside foundry'' tactics, which we will briefly outline in the following section.

In recent developments, researchers have proposed attacks that can be executed within the walls of an untrustworthy foundry environment [20], [22]. Unlike algorithmic SAT-based attacks, such as those mentioned in [15] and [21], these attacks do not rely on a functioning, activated chip as an oracle. While these attacks were originally designed with logic locking in mind, we posit that they are equally applicable to logic camouflaging schemes, as the two are mutually modelable. Additionally, the fundamental concept behind the attacks described in [20] and [22] revolves around the notion that an incorrect assignment of key bits leads to substantial logic redundancy when compared to a correct key-bit assignment. In [22], the analysis involves comparing the levels of logic redundancy for each logic value to deduce the probable values of individual key bits. Rangarajan et al. [33] have demonstrated the consequences of an erroneous key-input assignment, resulting in redundant logic. They have also proposed a solution in which the circuit's functionality varies at different stages.

In our research, we found the works of [20] and [22] to be pertinent. However, we refrain from making a direct, unbiased comparison as we lack access to these specific attack methods [20], [22]. To ensure a comprehensive security assessment, we carry out quantitative experiments based on the key findings referenced in the respective publications of [20] and [22]. Unlike the authors of [22], who reported success rates ranging from 25% to 75%, the *desynthesis*




(a) c432 (b) c499 (c) c880

(d) c1908 (e) c5315 (f) c6288

(g) c7552

**FIGURE 8.** OCR for different benchmarks for 50%, 70% and 90% correctly inferred polymorphic gates.

attack [20] is capable of accurately inferring 23 (up to 29) and 47 (up to 59) key bits for 32 and 64 key gates, respectively. To ensure a fair comparison, we consider similar ranges of correctly inferred gates.

To ensure a fair assessment, we camouflaged 45 signals for the ISCAS-85 benchmarks. The gates are chosen according to the algorithm 1. To replicate the attack results from [20] and [22], we use the following procedure. We write a script that selects the correct key input assignment among the hidden gates at random, yielding three sets that each correspond to 50%, 70%, and 90% of correctly inferred key inputs. Our camouflage scheme was built with Python scripts that ran on Verilog files. The program computes OCR using Cadence Xcelium with 10,000 input patterns, and functional correctness is determined using an SAT formulation to verify circuit equality [15].

We calculate the OCR for the encrypted netlist after calculating the percentage of correctly inferred gates for different levels of attack accuracy (50%, 70%, and 90%). The ISCAS-85 benchmark results are shown in Fig. 8. The OCR decreases as the proportion of correctly inferred gates increases, implying that the reconstructed netlist begins to behave similarly to the original circuit. We identified the 90% correctly inferred gates and ran it 25 times to ensure the correct OCR for different sets of input patterns, assuming a 90% attack accuracy for the ISCAS-85 benchmarks. 10,000 inputs were used each time. As a result, we determined OCR for a total of 25 × 10,000 input patterns for each encrypted benchmark. The average OCR for correctly inferred gates with 90%, 70%, and 50% accuracy is 10.87%, 22.11%, and 26.75%, respectively. These findings also suggest that "inside foundry" attacks can be avoided by masking a few carefully selected design signals.

### B. STUDY ON LARGE-SCALE IP PROTECTION AGAINST SAT-BASED ATTACKS

We concluded that the results shown in this section provide resiliency against untrusted end-users, which are closely followed in [15], [21], [23], and [24], as well as SoC integrator and design house levels.

The attacker has access to advanced, specialized tools for reverse engineering integrated circuits, such as setup for de-packaging and delayering ICs, imaging of individual layers, and image-processing tools.

- He or she can also tell the difference between a normal cell and one that has been camouflaged. When hybrid GSHE switches are used, the CMOS gates are easily identified. However, because hybrid shielding combines

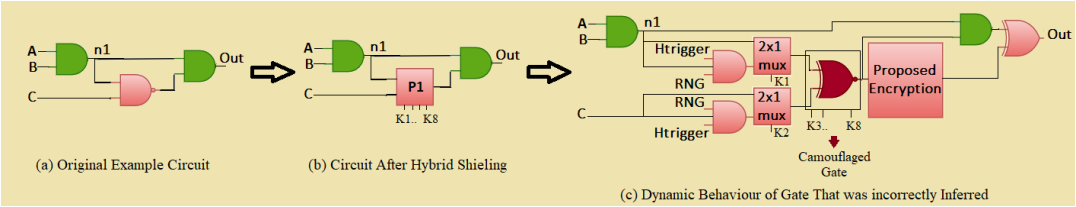



**FIGURE 9.** Effect of dynamic morphing.

logic encryption and camouflage, the complexity is greatly increased.

- The attacker is aware of the total number of hidden gates, as well as the variety and number of functions that each hidden cell performs.
- They acquire multiple copies of the chip from the open market, employing one as an oracle to observe the input-output relationships, and subsequently engage in reverse engineering of the remaining copies to retrieve the gate-level netlist of the chip. This process exposes vulnerabilities susceptible to SAT-based algorithms, as outlined in [15], [21], and [24].

Massad et al. [16] and Subramanyan et al. [15] independently demonstrated SAT-based attacks in 2015, offering security from logic camouflaging and logic locking, respectively. We perform a security analysis on an untrustworthy user by employing the easily accessible attack [15]. Then, we'll look at how dynamic camouflaging can help prevent attacks launched by malicious end users. We illustrate the related concept of run-time polymorphism. Look at the circuit in Fig. 9 for a conceptual representation of dynamic morphing.

Fig. 9 shows how dynamic camouflaging works. Part (a) of Fig. 9 depicts an original circuit. Part (b) depicts the circuit after hybrid shielding. Part (c) represents the dynamic behavior of the gate that was incorrectly inferred after the expansion of Part (b) insights. The dynamic behavior of an encrypted circuit with the same primary and key inputs is shown in table 3. Tables 3 show how dynamic behavior can perplex SAT attackers by changing the output without changing a primary or key input. Unknown to the user and random in behavior, the hidden trigger (Htrigger)[3] and random number generator (RNG)[4] can produce various results from the same primary input and key input. This behavior was referred to as dynamic morphing and was described in [33].

We used well-known SAT [15], Fault-Analysis [24], and Partial Break attacks to evaluate the proposed scheme, which uses GSHE switches to encrypt circuits. Initially, we tested our proposed schemes with 30 polymorphic gates and key input sizes ranging from 45, 60, and 90. In a few benchmarks, the scheme was found to be vulnerable to the SAT attack. Our method, on the other hand, is immune to all of the aforementioned SAT-based attacks when there are 45 or more polymorphic gates. Even after running them all for at least 48 hours, none of them decrypted at that time.

We also ran tests for the AppSAT attack [21]. The largest ISCAS 85 benchmarks, [19], c7552, were tested against AppSAT and proved to be resistant to the attack. We investigated various c7552 encrypted variants. We made the AppSAT cutoff by more than 24 hours. We assumed that if the tool did not provide an approximate key within the specified time, the attack would fail. In other words, we terminated the run after 24 hours and considered it a success for the proposed hybrid shielding.

When 30 polymorphic gates were encrypted using different key sizes, such as 45, 60, and 90, as shown in table 4, the scheme failed and the tool was able to approximate the key. The scheme passed the AppSAT test for 45 or more camouflaged gates, regardless of the size of the key inputs.

### C. PREVENTING FROM UNTRUSTED TEST FACILITY

Attackers in the testing facility with access to test patterns and corresponding output responses (created and provided by the reputable design house) may endanger the security assurances provided by LL and LC. Algorithms for Automatic Test Pattern Generation (ATPG) have been created to maximize fault coverage (FC) lowest test pattern count, which corresponds to a reduced test cost. However, such a strategy reveals crucial information on the specifics of the internal circuit [34].

In terms of VLSI testing principles, the detection of a stuck-at-fault involves two key elements: fault activation and fault propagation. A faulty node is assigned a value that is opposed to the fault that was induced on that node during fault activation.

1) A malicious employee in the testing facility, for example, could access the Gate-level camouflaged netlist obtained by RE.

[3] **Htrigger**:- We used any internal node at random as a Htrigger. As a result, it is impossible to reverse-engineer a specific structure. The user, on the other hand, has the option of selecting Htrigger. A trigger circuit is an alternative to Htrigger. In some cases, however, it may be possible to reverse engineer the trigger circuit design.

[4] **RNG Design**:- To make each input to the polymorphic gate dynamic, we must use a 2n bit RNG. A 2n-bit RNG, on the other hand, is easily disassembled and reverse-engineered. As a result, we've decided to split RNG into 2n sections. This means that we created 2n single-bit RNGs for every n polymorphic gate. Each polymorphic gate input has one. Using this technique, we were able to reduce the likelihood of the RNG being reverse-engineered. However, the unpredictability of dynamic behavior has been compromised in this case.

**TABLE 3.** Functional behaviour of Part (a), and (c) of Figure 9.

*Functional Behaviour of Part (a) of Figure 9*

| A | B | C | Out |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

*Dynamic Morphing Behaviour of Part (b) and (c) of Figure 9*

| A | B | C | RNG | Htrigger | K1 | K2 | Out |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | X | 1 | X | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | X | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | X | 0 |
| 1 | 1 | 0 | X | X | 0 | X | 0 |

**TABLE 4.** Results of proposed encryption approach to app-sat attack [21].

| Benchmark | KeySize/ Total Polymporphic Gates | Iterations | Time (hr) | AppSAT Resiliency |
|---|---|---|---|---|
| C7552 | 45/30 | 190 | 0.58 sec | Failed |
| C7552 | 60/30 | 1180 | 8.74 sec | Failed |
| C7552 | 90/30 | 520 | 2.54 sec | Failed |
| C7552 | 60/45 | 1950+ | 24+ | Pass |
| C7552 | 90/45 | 1950+ | 24+ | Pass |

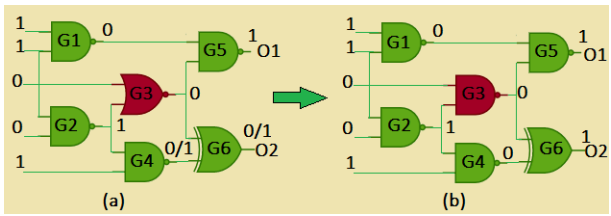



**FIGURE 10.** Hiding Test Results Using post-fabrication reconfigurability from the testing facility (a) A circuit in the test facility, NOR gate in Red is a wrongly inferred gate to provide security in the testing facility (b) A correct circuit with the correct assignment of the camouflaged gate in red.

2) He or she is also familiar with test infrastructure, such as locating scan chains and other components on the target chip.
3) The design firm has supplied both test patterns and their corresponding output responses. Additionally, the individual can leverage the ATPG tools employed in the generation of these test patterns.

Yasin et al. introduced the HackTest technique as documented in [34], which efficiently uncovers the actual nature of camouflaged gates through the utilization of test data. This attack capitalizes on the common practice of tailoring test pattern generation to achieve the highest fault coverage (FC). Consequently, an adversary can explore the key space, employing optimization methods to identify the most advantageous assignment of camouflaged gates that maximizes FC.

The internal characteristics of the underlying circuit, including gate types, their quantity, interconnections, and more, play a substantial role in the ATPG (Automatic Test Pattern Generation) process. Extensive research over the years has yielded robust algorithms that minimize the number of required test patterns while preserving a high fault coverage (FC). However, these algorithms currently do not incorporate security considerations, making test patterns a valuable resource for potential attackers. Nevertheless, due to post-fabrication reconfigurability, dynamic camouflaging has the capability to effectively thwart hack tests, as demonstrated in [33].

Figure 10(a) depicts the specially generated test pattern and the testing facility's respected output. Gate G3 is a cloaked gate that produces the incorrect output. NAND should have been used at first, as shown in figure 10(b). The input pattern 10(a) aids in identifying stuck-at-1 at node G4 by revealing its location. The "0/1" circuit output at O2 as 0 or 1, respectively, indicates a fault-free or faulty circuit. The test facility is equipped with the input pattern (11001) and anticipated output response (10) to test the produced ICs. The test data indicates that G3 is not a NAND and can be used to deceive an attacker. In figure 10(b), the correct assignment of the camouflaged gate is made using post-fabrication reconfigurability. This example shows how, in a testing environment, post-fabrication reconfigurability can effectively thwart an attacker. Please keep in mind that in this example, we only considered NAND and NOR operations for the camouflaged gate.

We conceal the same number of gates for the ISCAS-85 benchmarks to ensure a fair evaluation. We use the following strategy for experiments to demonstrate the system's resilience to HackTests [34]. We consider repeating this outside of our framework because [33] provides a thorough experiment demonstrating attack resistance for dynamic camouflaging. We'll show how, when applied to the same set of disguised gates, our method produces superior results. Gates are first chosen based on the algorithm 1 and then encrypted using hybrid shielding. We used camouflaged gates with 45 and 128 camouflaged signals to demonstrate attack resistance. For each benchmark, 250 and 60 key inputs were used for 128 and 45 camouflaged gates, respectively. We use the procedure below to replicate the attack results from [33]. Similarly to the previous section, we used a script to choose the correct assignment at random from among the hidden gates. This resulted in a set of benchmarks that each

**TABLE 5.** OCR results for 128 camouflaged gates using 250 keyinputs.

| Benchmark | OCR for 35.16% Correctly Inferred Gates | OCR for 0% Correctly Inferred Gate |
|---|---|---|
| C432 | 37.25% | 37.66% |
| C499 | 41.78% | 49.63% |
| C880 | 42.44% | 44.69% |
| C1355 | 40.15% | 41.24% |
| C1908 | 34.31% | 37.08% |
| C6288 | 46.53% | 46.42% |
| **% Average** | **40.41%** | **42.79%** |

**TABLE 6.** OCR results for 45 camouflaged gates using 60 keyinputs.

| Benchmark | OCR for 35.16% Correctly Inferred Gates | OCR for 0% Correctly Inferred Gate |
|---|---|---|
| C432 | 32.18% | 33.47% |
| C499 | 41.06% | 49.61% |
| C880 | 35.01% | 38.13% |
| C1908 | 25.96% | 28.37% |
| C5315 | 18.72% | 19.71% |
| C6288 | 34.09% | 34.89% |
| c7552 | 14.48% | 15.62% |
| **% Average** | **28.78%** | **31.4%** |

corresponds to 35.16% of correctly inferred gates taken from the [33], for ISACAS-85 benchmarks, which is an average of 45 correctly inferred gates out of 128 camouflaged gates. Table 5 displays the results for an average of 45 correctly inferred gates out of 128 correctly inferred gates, i.e. 35.16% and 0% correctly inferred gates, respectively. Even with 35.16% correctly inferred gates, the OCR is 40.41%, which is only 6.19% less than the original hybrid shielded circuits. We consider this a HackTest's success rate of 6.19%. This demonstrates that running the HackTest on hybrid shielded circuits has no discernible effect. In other words, the circuits are still behaving erratically, and the HackTest could only improve OCR accuracy by 6.19%.

Table 6 shows the results for 45 camouflaged gates with 60 key inputs. The results show that the OCR is 28.78% for 35.16% correctly inferred gates, which is only 8.34% less than the original hybrid shielded circuits. Even with 45 disguised gates, the HackTest efficiency is thus only 8.34%. When the results for the 128 and 45 camouflaged gates are compared, we find that the camouflaged gates increased by 64.85% while the OCR increased by 23.44%. As a result, even if we used 128 camouflaged signals to mask a large number of signals, no significant changes would occur in terms of OCR. Because the user can achieve maximum OCR with the fewest camouflaged gates while remaining resilient to multiple attacks, it exemplifies the benefit of using hybrid shielding.

#### 1) SENSITIVITY ATTACK AND MACHINE LEARNING ATTACK(SAIL)

For the stripped functionality locking schemes, the sensitivity analysis attack [23] can identify the protected input pattern. The attack method is divided into two parts: The locked circuit is pre-processed in the first part to identify the restoration unit by tracing the key inputs and then removing the restoration unit. This functionally strips the circuit's output. The average sensitivity of the functionality-stripped block is calculated in the second part. When compared to the other input patterns, it should be inverted for the protected input patterns. The proposed method is resistant to sensitivity attacks because no protected input can be extracted using sensitivity calculations. Furthermore, the proposed schemes have a high OCR, which means that the sensitivity of each primary output for any random set of primary inputs is very high. As a result, the hybrid shielding leaves no trace to calculate sensitivity for any set of input patterns.

Machine learning-based attacks (SAIL) [25] demonstrate how to recover a portion of the correct key with an accuracy rate of around 87%. However, as demonstrated in the previous section, the OCR is significant even for 90% correctly inferred gates, which is technically equivalent to 90% correct key assignment. which demonstrates the utility of dynamic camouflage in conjunction with logic locking. As a result, even a small number of dynamically camouflaged signals can render machine-learning attacks ineffective.

#### 2) PREVENTING REVERSE ENGINEERING AND SIDE-CHANNEL ATTACKS

The GSHE switch's layout is uniform [10], making it indistinguishable from optical-imaging-based-RE. The primitive is polymorphic, demonstrating its ability to camouflage internal nodes. As a result, a RE-centric attacker is likely to misinterpret parts of the layout. On a full-chip scale, resolving all dynamic features at the same time is nearly impossible.

CMOS devices emit photons while operating, making them vulnerable to sophisticated attacks like [35]. The GSHE switch itself emits no photons [10]. Because of the fundamentally different switching principle, the proposed algorithm with the GSHE switch is inherently resistant to photon-based read-out attacks.

## V. CASE STUDY I: WHY STABLE SIGNALS?

A major concern has always been the selection of internal signals that are highly suitable for camouflage. We present a novel method for selecting internal signals that provide high OCR and resilience against a variety of attacks at various IP supply chain levels in this paper. For camouflage, we recommended selecting a node with a high weight and stability. When compared to random signal selection, the proposed method significantly improved OCR. Using a simple example, we can better understand the reasoning behind selecting highly stable nodes.

### A. EXAMPLE

We discovered that G4 was the most stable node and G3 was the most sensitive when the algorithm 1 was applied to the circuit diagram in figure 11. The outputs for each input pattern are listed in table 7. Table 7 lists the respected outputs for randomly generated input vectors. According to

**TABLE 7. Functional behaviour of Figure 11.**

(a) Original Functionality of Figure 11

| A | B | C | D | G3 | G4 | O1 | O2 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

(b) Original Functionality with Random Input Patterns of Figure 11

| A | B | C | D | G3 | G4 | O1 | O2 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

(c) Circuit Output After Camouflage Most Sensitive Node in Figure 11

| A | B | C | D | G3' | G4 | O1 | O2 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

(d) Circuit Output After Camouflage Most Stable Node in Figure 11

| A | B | C | D | G3 | G4' | O1 | O2 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

table 7, after input vectors are randomized, G3 is the most sensitive node and G4 is the most stable node. In this case, the probability of selecting G3 as an appropriate gate for any sequence of randomized input vectors is 100% because, even if G3 and G4 have equal stability, the algorithm will select G3 as an appropriate gate due to its larger weight. The table 7 shows how output changed after camouflage G3. The overall OCR in this case is only 6.25%. Remember that the camouflaged gate can perform the boolean operations XOR and XNOR and that for each input pattern, we assume incorrect key input.

Results were obtained similarly when we camouflaged G4, as shown in table 7. The overall OCR for this case is 65.63%, which is 9.52 times higher than the other case. Please keep in mind that for each input pattern, we assume incorrect key input, and camouflage gates can only perform AND and OR Boolean operations.

## VI. CASE STUDY II: WHY 45 STABLE SIGNALS?

Instead of choosing highly stable signals, we calculate the OCR and SAT resiliency for gradually increasing the % of random signals to camouflage. With the help of this

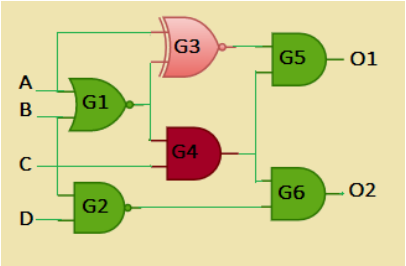



**FIGURE 11.** Effect on Outputs of Proposed Scheme to Select Nodes: G4 is the most stable node and G3 is the most sensitive node.

**TABLE 8.** OCR and SAT resiliency results for 30 camouflaged gates using 45 keyinputs.

| Benchmark | % of Random Signals | SAT Resiliency | OCR in % |
|---|---|---|---|
| C499 | 0% | 136.376 sec | 39.21% |
| C499 | 5% | 483.033 sec | 37.59% |
| C499 | 10% | 429.64 | 37.40% |
| C499 | 15% | 506.334 | 33.09% |
| C499 | 20% | 20520.5 | 33.02% |
| C499 | 25% | 23937.7 | 32.14% |
| C499 | 50% | - | 27.55% |
| C499 | 75% | - | 20.17% |
| C499 | 100% | - | 17.35% |

study, we can track changes in OCR as we modify a certain percentage of randomly chosen nodes. Two conclusions can be drawn from the data presented in table 8. First, OCR is only 17.35% for 100% randomly camouflaged signals while it is 39.21% for 0% randomly camouflaged signals. Second, as the proportion of random signals used for camouflage increases, SAT resilience rises while OCR falls. Hence, both are inverse to each other. It should be noted that if the percentage of random signal selection is high, sat resilience can be attained even with a low number of camouflage gates, but OCR will be compromised. In this instance, we only used 30 camouflaged gates to achieve SAT resilience, but the OCR is compromised by 29.74%; these results use a 50% random signal selection. The difference is even greater, with an OCR decline of 55.75% for 100% random signal selection as compared to the suggested algorithm.

## VII. PPA CALCULATION

The calculation of PPA overhead is a critical aspect of evaluating proposed encryption methods. However, the literature lacks a method for quantifying the PPA overhead specific to hybrid logic encryption techniques. In this paper, we introduce a mathematical framework to estimate the approximate PPA for circuits encrypted with hybrid shielding. Our approach involves a two-step process. Firstly, we apply logic encryption to the source circuit without substituting selected signals with polymorphic gates. Once the circuit is encrypted, we leverage Electronic Design Automation (EDA) tools to compute the PPA overhead. It is important to note that these extracted PPA values do not represent the complete PPA overhead of hybrid shielding. The second step entails determining the PPA overhead introduced by dynamic GSHE-based polymorphic gates, for which we employ the values (Area: 0.029 um2, Power: 0.2673 uW, Delay: 1.83 ns) provided in [29]. By combining the PPA values of the encrypted circuit and the PPA values associated with the dynamic GSHE-based polymorphic gates using a prescribed formula, we derive the final PPA values for the hybrid shielding-encrypted circuits, as presented in Table 9.

**TABLE 9.** PPA overhead for hybrid shielding for 45 polymorphic gates.

| Benchmark | Power Overhead ($uW$) | Delay Overhead ($ns$) | Area Overhead ($um^2$) |
|---|---|---|---|
| C432 | 299.85 | 5.94 | 1555.78 |
| C499 | 895.97 | 6.47 | 2658.34 |
| C880 | 465.21 | 5.62 | 2557.02 |
| C1908 | 979.41 | 7.37 | 2599.97 |
| C5315 | 1684.31 | 4.99 | 6426.93 |
| C6288 | 5545.2 | 14.28 | 9655.1 |
| C7552 | 2494.31 | 6.78 | 8267.06 |

### A. TOTAL AREA CALCULATION

The below formula provides a concise representation of the total area calculation for hybrid shielding-encrypted circuits, accounting for both the original circuit area and the additional area introduced by polymorphic gates while subtracting the areas of replaced gate cells.

$$\text{Total Area} = \text{A}_{\text{enc}} + (\text{A}_{\text{polywmux}} \cdot n) - \left(\sum \text{A}_{\text{replaced gates}}\right)$$

$\text{A}_{\text{enc}}$: Represents the total area of the encrypted file without polymorphic gates. $\text{A}_{\text{polywmux}}$: Signifies the total area of one polymorphic gate with MUXs, multiplied by the total number of polymorphic gates ($n$). $\sum \text{A}_{\text{replaced gates}}$: Encompasses the collective area of all replaced gate cells (e.g., AND gates, NAND gates, etc.), calculated by subtracting their respective areas from the total area.

### B. TOTAL POWER CALCULATION

The below formula provides a comprehensive representation of the total power calculation for hybrid shielding-encrypted circuits, accounting for both the original circuit's power consumption and the additional power introduced by polymorphic gates while subtracting the power contributions of replaced gate cells.

$$\text{Total Power} = \text{P}_{\text{enc}} + (\text{P}_{\text{polywmux}} \cdot n) - \left(\sum \text{P}_{\text{replaced gates}}\right)$$

$\text{P}_{\text{enc}}$: Represents the total power consumption of the encrypted file without polymorphic gates. $\text{P}_{\text{polywmux}}$: Signifies the total power consumption of one polymorphic gate with MUXs, multiplied by the total number of polymorphic gates ($n$). $\sum \text{P}_{\text{replaced gates}}$: Encompasses the collective power consumption of all replaced gate cells (e.g., AND gates, NAND gates, etc.), calculated by subtracting their respective power contributions from the total power.

### C. TOTAL DELAY CALCULATION

The below formula offers a comprehensive representation of the total delay calculation for hybrid shielding-encrypted

circuits. It factors in the original circuit's delay, the delay introduced by polymorphic gates, and subtracts the delay contribution of the replaced gate cell to yield the overall delay.

$$\text{Total Delay} = \text{Delay}_{\text{enc}} + \text{Delay}_{\text{polywmux}} - \left(\sum \text{Delay}_{\text{replaced gate}}\right)$$

$\text{Delay}_{\text{enc}}$: Represents the total delay of the encrypted file without polymorphic gates. $\text{Delay}_{\text{polywmux}}$: Signifies the delay introduced by one polymorphic gate with MUXs. $\sum \text{Delay}_{\text{replaced gate}}$: Encompasses the delay of the replaced gate cell, calculated by subtracting their respective delay contribution from the total delay.

## VIII. CONCLUSION

We present a novel netlist-level hybrid shielding scheme that combines logic encryption and logic camouflage. We also presented a novel method for selecting internal signals to achieve high OCR. We have provided numerous examples of why the proposed algorithm is superior to random signal selection. This method is resistant to SAT attacks, AppSAT attacks, Sensitivity attacks, HackTests, Key-Sensitization attacks, Fault Analysis attacks, and many other types of attacks. In most cases, a camouflaged internal state results in a higher rate of output corruption than random encryption. The same level of corruption cannot be achieved by any existing SAT resilient schemes supplemented with random encryption of up to 30 additional key inputs. The general trend is for more polymorphic gates to have a high corruption rate. It is also beneficial to SAT resiliency. The results show that encrypted circuits with 45 or more polymorphic gates are SAT resilient. Hence, it is recommended to use at least 45 polymorphic gates when encrypting a design.

## REFERENCES

[1] K. Shamsi, M. Li, K. Plaks, S. Fazzari, D. Z. Pan, and Y. Jin, "IP protection and supply chain security through logic obfuscation: A systematic overview," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 24, no. 6, pp. 1–36, Sep. 2019.

[2] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 709–720.

[3] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, Oct. 2010.

[4] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT attack on logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 2, pp. 199–207, Feb. 2019.

[5] 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 Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1601–1618.

[6] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2016, pp. 236–241.

[7] Y. Liu, M. Zuzak, Y. Xie, A. Chakraborty, and A. Srivastava, "Strong anti-SAT: Secure and effective logic locking," in *Proc. 21st Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2020, pp. 199–205.

[8] N. Saxena, R. V. Narayanan, J. K. Meka, and R. Vemuri, "SRTLock: A sensitivity resilient two-tier logic encryption scheme," in *Proc. IEEE Int. Symp. Smart Electron. Syst. (iSES)*, Dec. 2021, pp. 389–394.

[9] N. Saxena and R. Vemuri, "ISPLock: A hybrid internal state locking method using polymorphic gates," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2022, pp. 140–145.

[10] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Advancing hardware security using polymorphic and stochastic spin-Hall effect devices," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 97–102.

[11] M. Yasin, J. J. Rajendran, and O. Sinanoglu, *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Berlin, Germany: Springer, 2020.

[12] I. R. Nirmala, D. Vontela, S. Ghosh, and A. Iyengar, "A novel threshold voltage defined switch for circuit camouflaging," in *Proc. 21st IEEE Eur. Test Symp. (ETS)*, May 2016, pp. 1–2.

[13] B. Erbagci, C. Erbagci, N. E. C. Akkaya, and K. Mai, "A secure camouflaged threshold voltage defined logic family," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2016, pp. 229–235.

[14] S. Patnaik, M. Ashraf, O. Sinanoglu, and J. Knechtel, "Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4466–4481, Dec. 2020.

[15] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2015, pp. 137–143.

[16] M. E. Massad, S. Garg, and M. Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–14.

[17] S. Datta, S. Salahuddin, and B. Behin-Aein, "Non-volatile spin switch for Boolean and non-Boolean logic," *Appl. Phys. Lett.*, vol. 101, no. 25, Dec. 2012, Art. no. 252411.

[18] Y. Zhang, B. Yan, W. Wu, H. Li, and Y. Chen, "Giant spin Hall effect (GSHE) logic design for low power application," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2015, pp. 1000–1005.

[19] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Mar. 1985, pp. 677–692.

[20] M. El Massad, J. Zhang, S. Garg, and M. V. Tripunitara, "Logic locking for secure outsourced chip fabrication: A new attack and provably secure defense mechanism," 2017, *arXiv:1703.10187*.

[21] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2017, pp. 95–100.

[22] L. Li and A. Orailoglu, "Piercing logic locking keys through redundancy identification," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 540–545.

[23] J. Sweeney, M. J. H. Heule, and L. Pileggi, "Sensitivity analysis of locked circuits," in *Proc. 23rd Int. Conf. Logic Program., Artif. Intell. Reasoning*, vol. 73, 2020, pp. 483–497.

[24] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. DAC Design Autom. Conf.*, Jun. 2012, pp. 83–89.

[25] P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine learning guided structural analysis attack on hardware obfuscation," in *Proc. Asian Hardw. Oriented Secur. Trust Symp. (AsianHOST)*, Dec. 2018, pp. 56–61.

[26] S. Anceau, P. Bleuet, J. Clédière, L. Maingault, J. L. Rainard, and R. Tucoulou, "Nanofocused X-ray beam to reprogram secure circuits," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2017, pp. 175–188.

[27] J. Wang, C. Shi, A. Sanabria-Borbon, E. Sánchez-Sinencio, and J. Hu, "Thwarting analog IC piracy via combinational locking," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2017, pp. 1–10.

[28] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proc. IEEE*, vol. 101, no. 12, pp. 2498–2533, Dec. 2013.

[29] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Spin-orbit torque devices for hardware security: From deterministic to probabilistic regime," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 8, pp. 1591–1606, Aug. 2020.

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

[31] N. Rangarajan, A. Parthasarathy, N. Kani, and S. Rakheja, "Energy-efficient computing with probabilistic magnetic bits—Performance modeling and comparison against probabilistic CMOS logic," *IEEE Trans. Magn.*, vol. 53, no. 11, pp. 1–10, Nov. 2017.

[32] J. C. Slonczewski, "Current-driven excitation of magnetic multilayers," *J. Magn. Magn. Mater.*, vol. 159, nos. 1–2, pp. 1–7, Jun. 1996.
[33] N. Rangarajan, S. Patnaik, J. Knechtel, R. Karri, O. Sinanoglu, and S. Rakheja, "Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 137–156, Jan. 2022.
[34] M. Yasin, O. Sinanoglu, and J. Rajendran, "Testing the trustworthiness of IC testing: An oracle-less attack on IC camouflaging," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2668–2682, Nov. 2017.
[35] A. Schlosser, D. Nedospasov, J. Kramer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES," in *Cryptographic Hardware and Embedded Systems—CHES*, E. Prouff and P. Schaumont, Eds. Berlin, Germany: Springer, 2012, pp. 41–57.

**NIKHIL SAXENA** was born in Gwalior, Madhya Pradesh, India, in 1988. He received the B.E. degree in electronics and communication engineering from Rajiv Gandhi Technical University, in 2010, and the M.Tech. degree in microelectronics from the Jaypee Institute of Information Technology, in 2013. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Cincinnati. He has recently accepted a position as a Lecturer with the University of Massachusetts Amherst. As part of his Ph.D. research, he was a Research Assistant with the Digital Design Environments Laboratory. He wrote this work while pursuing the Ph.D. degree with the University of Cincinnati. He was an Assistant Professor with the Electronics and Communication Department, Institute of Technology and Management, for six years. During his time as an assistant professor, he has published over 30 papers in journals and international conferences. He has graduated 17 master's degree candidates. His research interests include hardware security, logic encryption, VLSI, and design automation. He received the UC Robert J. Herbold Fellowship, in 2022.

**RANGA VEMURI** (Senior Member, IEEE) has been with the Faculty of Electrical and Computer Engineering, University of Cincinnati, since 1989, where he is currently a Professor. He and his students have published over 300 articles and received 13 best paper awards or nominations. He graduated over 40 Ph.D. and 80 M.S. students. His research has funded by AFRL, DARPA, NSF, SRC, State of Ohio, and various industries, including Edaptive Computing Inc., to the tune of $12M over the past 25 years. His research interests include hardware security and trust, correctness and security, VLSI design and architectures, embedded systems, formal methods and formal verification, electronic design automation, logic and physical synthesis, and reconfigurable computing. He was an Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and a Guest Editor of the IEEE COMPUTER.