# ENTANGLE: An Enhanced Logic-locking Technique for Thwarting SAT and Structural Attacks

Armin Darjani
Technische Universität Dresden
Dresden, Germany
armin.darjani@tu-dresden.de

Nima Kavand
Technische Universität Dresden
Dresden, Germany
nima.kavand@tu-dresden.de

Shubham Rai
Technische Universität Dresden
Dresden, Germany
shubham.rai@tu-dresden.de

Mark Wijtvliet
Technische Universität Dresden
Dresden, Germany
mark.wijvliet@tu-dresden.de

Akash Kumar
Technische Universität Dresden
Dresden, Germany
akash.kumar@tu-dresden.de

## ABSTRACT

Among the SAT-resilient logic locking techniques, the Stripped-Functionality-Logic-Locking (SFLL) is the most promising solution which can guard the intellectual property against approximate, sensitization, SAT, and structural attacks which target Point-function techniques. However, even the SFLL technique has been shown to be vulnerable to a recent class of structural attacks that identify the perturbation logic. In this paper, we first categorize all possible classes of attacks on SFLL. Then we propose ENTANGLE , a novel logic locking technique built upon SFLL that can resist all of these attacks, including the emerging ML-Based attacks. We test our technique against publicly available SFLL attacks. The implementation results show that ENTANGLE can secure large-sized industrial circuits with an average overhead of 11.6 percent and 9.1 percent for area and power, respectively.

## CCS CONCEPTS

• **Security and privacy → Security in hardware**.

## KEYWORDS

Structural attack, Design-for-trust, Reverse engineering, Logic locking, Boolean satisfiability (SAT)

**ACM Reference Format:**
Armin Darjani, Nima Kavand, Shubham Rai, Mark Wijtvliet, and Akash Kumar. 2022. ENTANGLE: An Enhanced Logic-locking Technique for Thwarting SAT and Structural Attacks. In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3526241.3530371

## 1 INTRODUCTION

Logic locking is a holistic design-for-trust technique that can shield the IP through different stages of the IC supply chain. This technique locks the design by adding new logic elements to the circuit, hiding the true functionality of the design. The locked design only functions correctly upon receiving a true set of key bits stored in a tamper-proof on-chip memory. The preliminary logic locking techniques [3] [4] [14] utilized XOR/XNOR gates to bring the most corruption to the output of the circuit if the wrong key is in place. However, these early approaches were broken by Boolean satisfiability (SAT)-based attack [12]. To overcome this powerful attack, two new classes of logic locking techniques emerged. The first class is SAT-hard techniques that insert structures in a netlist that are hard to resolve for a SAT solver [5]. The second class is Point-function based techniques [15] [2] that maximizes the number of clauses of the SAT attack.

Figure 1 is an overview of the Point-function techniques that contain an original design unit and a protection unit defined with blue color. The primary inputs (PIs) of the design unit are divided into two protected ($X_{protected}$) and unprotected ($X_{unprotected}$) sets. Here the protected set is fed to the Point-function along with the key set (K) containing key inputs (KIs) to flip the design output upon arriving particular combinations of the key and protected input. If the protected inputs are equal to the key inputs, the Point-function outputs one that flips the output of the design. In these circuits, the attack would need $2^k + 1$ different input patterns to break the circuit. In these techniques, the correct key is hard-coded in the Point-function.

These techniques are vulnerable to approximate and structural attacks [8] [10]. As the name suggests, the aim of approximate attacks is to find a key pattern that can unlock the biggest functional subset of the circuit (an approximate circuit). Moreover, these techniques leave the original design (design unit in Figure 1) without any changes that expose the circuit to the structural [17] attacks.

To overcome the approximate attack, one can intertwine these techniques with one of the early logic locking techniques like SLL (Strong logic locking) [14]. However, variants of SAT-attack like AppSAT [8], and DIP [10] can break these methods. In structural attacks, the attacker can follow the transitive fan-out cone of KIs to identify the protection blocks and, ultimately, the output of the Point-function unit that corrupts the output of the circuit. Note that the fundamental flaw of the Point-function techniques is rooted in the complete separation of the protection circuit and the original design.

Stripped-functionality-logic-locking (SFLL) [16] is a powerful tunable technique that can address both the approximate and structural attacks which target Point-function techniques. By stripping the functionality of the original design for some input patterns and restoring the original functionality using the restore unit, the SFLL compensates for the main weakness of the Point-function techniques. However, the perturb unit leaves particular structural

*GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA.*
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9322-5/22/06…$15.00
https://doi.org/10.1145/3526241.3530371

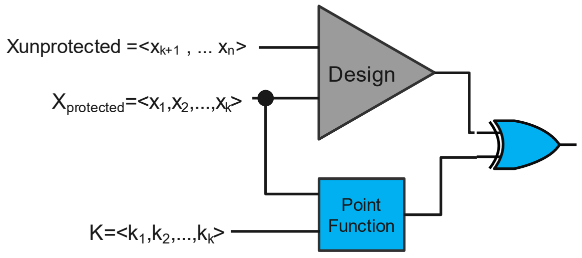


**Figure 1: Point-function Techniques Overview**

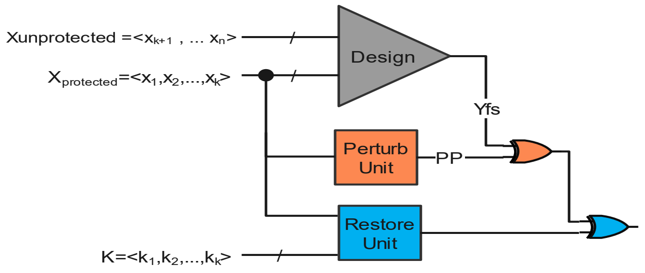


**Figure 2: SFLL technique overview**

and logical traces. Consequently, emerging netlist analysis-based attacks [13] [11] [1] identify the add-on logic, and enable the attacker to circumvent the protection logic.

**Contributions**: Considering the above discussion, a secure locking technique that can guard the circuit against all SAT, approximate and structural attacks is demanding. The contributions are:

(1) We introduce ENTANGLE that minimizes the structural and logical traces of the perturb unit without relying on the security-agnostic EDA tools.
(2) To further fortify ENTANGLE, we present two tunable cloaking methods that can hide the perturb unit by differentiating the transitive fan-in cone of the restoration and perturb units.
(3) We compare ENTANGLE with the original SFLL regarding the security properties against SAT, approximate, and structural attack using state-of-the-art attacks [13] [11] [1].

## 2 PRELIMINARIES

The goal of an SFLL technique is to modify the original design for some input patterns that are hamming distance $h$ from the protected pattern using an add-on protection unit called perturb. The protected pattern contains a subset of PIs. Only upon placing the right key in the restore unit, the original functionality of the circuit will be restored. This way, SFLL becomes resilient against removal attacks. Figure 2 shows an overview of the SFLL technique. This protected circuit contains design (gray), perturb (orange), and restore (blue) units. Like the Point-function technique, the PIs of the design unit are divided into two protected ($X_{protected}$) and unprotected ($X_{unprotected}$) sets. The protected set is fed to both perturb and restore units. The key set (K) containing KIs is only fed to the restore unit, which outputs one if the KIs are hamming distance $h$ from the PIs coming from the protected set.

The perturb unit outputs one (PP=1) for all the PIs that are hamming distance $h$ from the protected pattern that is hard-coded in this unit. When $PP = 1$, the output of the original design (Yfs) is flipped by the perturb unit. The restore unit can correct these perturbations if the KIs are equal to the protected pattern. So, in the SFLL, the correct key is equal to the protected pattern. The final output of the circuit is incorrect for $\binom{k}{h}$ combinations of KIs for each PIs combination where k is the number of key bits[16]. Here inferring the key from the restore unit would be impossible as the protected pattern is hard-coded in the perturb unit. A designer can leverage hamming distance value to bring a trade-off between SAT and approximate attacks based on the security needs that give an edge to the SFLL over other post-SAT logic locking techniques.

### 2.1 Threat Model

Before discussing the SFLL weaknesses, we first have to clarify our assumptions about the attacker. We assume the attacker has access to the gate-level locked netlist, an unlocked chip (oracle), technology library, hamming distance in SFLL-HD, and can distinguish between primary inputs (PIs) and key inputs (KIs).

### 2.2 SFLL Fundamental Structural Weaknesses

Considering the threat model, the attacker can utilize the following weaknesses to initiate and carry out an attack:

1) The main structural weakness of the SFLL roots in using the same set of PIs in both restoration and perturb units. By tracing the KIs and finding the transitive fan-in cone of the restoration unit, all the PIs participating in a protected pattern can be found. Moreover, the tip of the restoration unit cone is connected to the protected output. The attacker can utilize this knowledge to prune the cone of the protected output and find the signals fed only by the previously found PIs. Some of these signals will be the signals related to the perturb unit.

2) Perturb unit uses a hamming distance checker to strip the output. Knowing the value of the hamming distance and the PIs relevant to the protected pattern, the attacker can search for a structure that looks like a hamming distance checker. This unit usually has a symmetrical and tree-like structure.

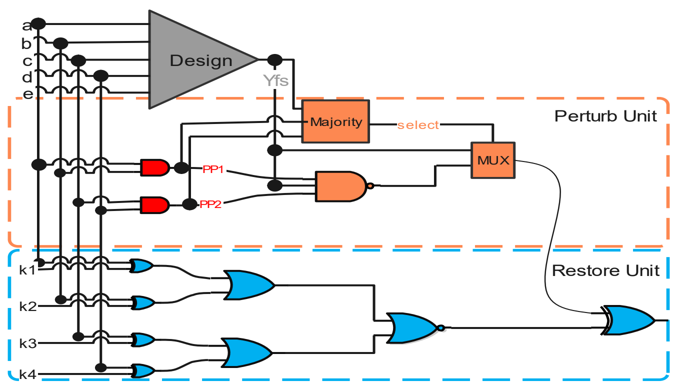
3) Each key is XORed/XNORed with a particular PI from the protected set. so if the attacker can find the protected pattern, each bit of the correct key is equal to its relevant PI in the restore unit.

4) *PP* has some logical properties like low activity that can be harnessed for pruning suspicious signals.

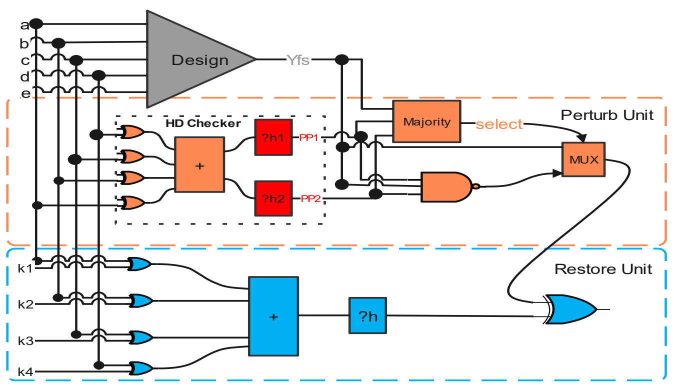
### 2.3 Attacks on SFLL

The structural flaws of the naive implementation of the SFLL have guided researchers to propose netlist-analysis based attacks[13] [11] [1]. We categorize the existing and anticipated attacks into two classes.

**Class A**: The first class is algorithmic attacks. The main goal of these attacks is to find the perturb signal (PP) and utilize it to extract the correct key. All these attacks start with finding PIs in the protected pattern set and the protected output of the circuit by analyzing the transitive logic cone of the restore unit. Then, the algorithms search for signals in the logic cone of the protected output. If it finds a signal that its transitive fan-in contains all the PIs in the protected set and no other PIs, it marks it as a suspicious signal. One of these suspicious signals is the perturb (*PP*) signal. The algorithms continue with pruning these suspicious signals knowing the properties of the perturb signal. Finally, the algorithms use SAT solvers to find the input patterns that set the perturb signal to logic one and infer the key from these input patterns.



(a) ENTANGLE -TTLock



(b) ENTANGLE -HD

Figure 3: ENTANGLE

**Class B**: The second class is ML-based attacks that harness the power of graph neural networks (GNNs) to classify the circuit gates. GNNUnlock [1] utilizes GNN to learn the structural features of the add-on protection units. Then it classifies the gates into three classes, perturb, restoration, and design. After sorting out the misclassifications using a post-processing algorithm, the original design gates would be separated from the protection-related gates. Finally, the attacker searches for the output of the perturb and restore units separately. Putting these signals to constant value zero, the attacker can circumvent the SFLL.

## 3 ENTANGLE

**Motivation**: The naive implementation of the SFLL that relies only on the security-agnostic EDA tools to hide the perturb unit is vulnerable to all classes of attacks. To overcome the structural weakness of the SFLL, [6] was proposed. This technique utilizes fault injection techniques to break the structural traces of the perturb unit; however, it brings an infeasible LUT footprint that is impractical to implement. Moreover, it cannot protect a large number of patterns as SFLL-HD. The authors in [7] tried to compensate the LUT footprint; nevertheless, their technique is only as powerful as TTLock that is vulnerable to simple approximate attacks.

We present ENTANGLE that retains the security properties of the SFLL and eradicates the reliance on security-agnostic tools to blend the perturb unit from the attackers.

### 3.1 Design and analysis

ENTANGLE breaks the perturb signal into two signals coming from separate cones and then entangles them with the output of the original circuit. Figure 3 shows the ENTANGLE-TTLock and -HD. In both these designs, the protected pattern is set to logic *<1,1,1,1>*. In ENTANGLE-TTLock, the last stage of the AND-tree is removed, leaving two separate AND-tree (red-colored ANDs) with different PIs in their logic cones. For ENTANGLE-HD the value of the hamming distance is separated into *h1*, and *h2* bits and each set of bits is compared with the corresponding bits of the intended hamming distance. If both the *PP1* and *PP2* signals are equal to logic 1, the perturb unit should flip the Yfs, which is the output of the original circuit. In other cases, the output of the perturb unit should be the Yfs itself.

ENTANGLE preserves all the security properties of the SFLL against SAT and approximate attacks as the functionality remains the same. In the following, we analyze the security promises of the ENTANGLE against two classes of attacks introduced in Section 2.3.

**Class A**: The algorithmic attacks search for special signals that contain all and only the protected PIs (PIs in the protected set). In ENTANGLE-TTLock, each *PP1* and PP2 contains just a portion of the protected PIs, so the algorithms do not mark these signals.

If the hamming distance is greater than zero ($HD^{>0}$) the attacks can mark both *PP1* and *PP2* as suspicious signals. Nevertheless, each of these signals can be set to logic 1 with combinations of the protected set that will set the original *PP* signal to zero. As a result, the algorithms prune these signals from the suspicious list. For example, the SlidingWindow [11] algorithm in the FALL attack finds distinct satisfying assignments for the suspicious signal. Then it infers that if the two satisfying assignments are of hamming distance *2h* then the equal bits in both patterns are equal to corresponding key bits. It then adds the new clause to the SAT query. Because in ENTANGLE each of the *PP1* and *PP2* contains part of the protected pattern, they will be pruned like other suspicious signals as the SAT query will contain inconsistent clauses [11]. The same goes for all other algorithmic attacks, including SFLL-HD-Unlocked. They ultimately come up with some inconsistent clauses that force the algorithm to prune the *PP1* and *PP2*.

**Class B**: An ML-based attack aims to classify the design nodes (logic gates) to remove the nodes relevant only to the protection unit(s). GNNUnlock classifies the circuit nodes into the design, perturb and restore nodes utilizing graph neural networks. To rectify the misclassifications, it utilizes a post-processing algorithm.

In the case of ENTANGLE-TTLock the final stage is broken into two AND-trees. Each of these trees contains an arbitrary number of the PIs from the protected set. Moreover, the points of connection between these two AND-trees are the majority unit and the NAND gate containing Yfs as one of their inputs. So, there will be no node in the design with a transitive fan-in cone containing only all the perturb nodes and protected PIs. Moreover, the structural traces of the AND-trees can be broken using techniques like DTL [9] that further obfuscate the perturb unit.

Like ENTANGLE-TTLock, finding the tip of the perturb unit in ENTANGLE -$HD^{>0}$ is not possible as the *PP1* and *PP2* signals are entangled with the Yfs. However, unlike the ENTANGLE-TTLock where the structural traces of the perturb unit could be cloaked entirely, ENTANGLE-$HD^{>0}$ leaves traces in *PP1* and *PP2* signals. The hamming distance checker is composed of a set of XOR gates, an adder unit, and a comparator. Also, the ENTANGLE-$HD^{>0}$ breaks the comparator structure; the adder structure containing all protected PIs still exists in the design. So a well-trained GNN may still recognize almost all the perturb nodes. To further obfuscate the ENTANGLE-$HD^{>0}$, we propose two cloaking techniques.

## 3.2 Cloaking techniques

**Dummy logic**: This technique brings some PIs to the perturb unit that are not incorporate in the protected set that can weaken both classes of attacks. Figure 4 shows this cloaking unit. It adds one hamming distance checker and two Multiplexers to the HD checker unit of the ENTANGLE. Signal *DS* is formed by arbitrary nodes from the design unit. The only limitation is that these arbitrary nodes must contain at least one PI from the unprotected set in their transitive fan-in cones (like *e* in Figure 3). This signal blends with the protected PIs at the adder unit of the hamming distance checker. If the output of the adder is *h+1*, then the output of the HD checker should be equal to a set of two elements {DS,DS}. Otherwise, the output of this logic is equal to the default HD checker of the ENTANGLE.

This technique can misguide the post-processing of the class B attacks. The post-processing algorithm in [1] checks all the classified nodes and if one of the nodes marked as perturb contains any PI other than the protected set members in their transitive fan-in cone, it reclassifies it as a design node. So this technique misleads the post-processing algorithm to misclassify some of the perturb nodes connected to the HD checker unit in Figure 3.

**Dummy key bit**:Dummy key bit conceals the perturb unit by connecting PIs from the unprotected set to the restore unit using a dummy KI. Figure 5 shows an example of this technique. Here the inner logic contains some arbitrary logic elements containing exactly one PI from the unprotected set. Feeding the cloaking unit with the correct dummy key bit must set the output of the cloaking unit to zero. This way, it does not affect the hamming distance checker. Since the add-on unprotected PIs are connected to new KIs, the attacker cannot distinguish them from PIs coming from the protected set.

The above two techniques can be used in conjunction to misguide the post-processing algorithm. Consider set $S$ is a set of the arbitrary nodes that form the *DS* signal in the dummy logic technique. All these nodes will be misclassified as perturb nodes by the GNN. Now, consider these arbitrary nodes have a set of $U$ PIs from the unprotected set. If all the members of $U$ are connected to the restore unit using dummy keys, the post-processing algorithm cannot correct this misclassification. Moreover, the other design nodes that are only connected to the members of the $S$ will be misclassified as perturb nodes by the post-processing algorithm.

The dummy key bit only affects the restore unit. To analyze the effects of this technique on the approximate attack, the OER (output error rate) of the ENTANGLE for each incorrect key should be considered. This OER can be calculated based on hamming distance, number of key bits, and number of dummy key bits.

For examining the SAT resiliency of the ENTANGLE after adding dummy key bits, we have to compare it with the case that we add the same number of real key units (a KI XORed with a PI). If we add $\alpha$ number of real key units to the restore unit, each pattern

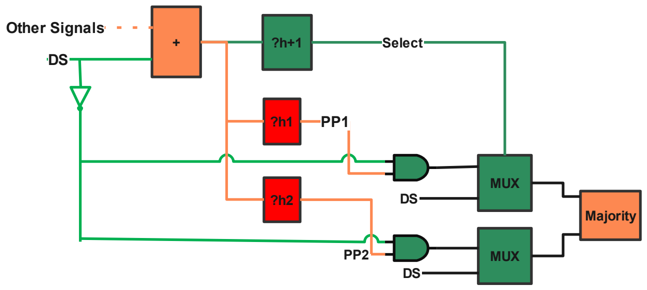


Figure 4: Dummy Logic Technique

can reveal $\binom{k+\alpha}{h}$ incorrect keys in each SAT iteration. In our design, when adding dummy keys, the worst case happens when all the inner logic elements of the cloaking units output ones. In this case, we can calculate the number of incorrect keys that SAT attack finds in each iteration (1). This number can be equal, worse, or better than the case of adding real key units based on the values of k, h, and $\alpha$.

$$\binom{k}{h} + \binom{k}{h-1} + \dots + \binom{k}{h-\alpha} \tag{1}$$

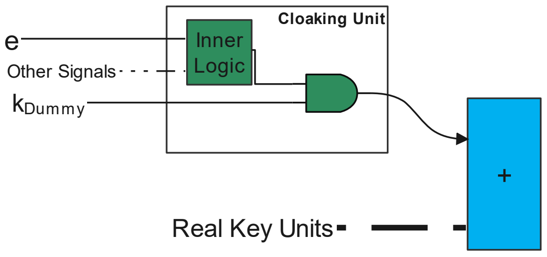


Figure 5: Dummy Key Bit Technique

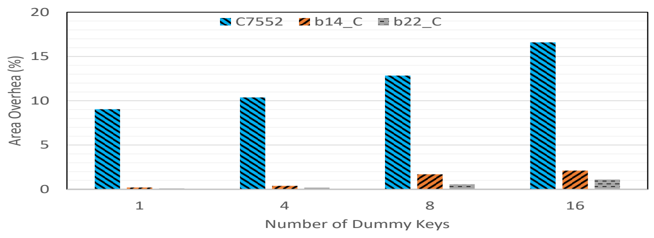
## 4 EXPERIMENTAL RESULTS

## 4.1 Experimental Setup

For the evaluation of the ENTANGLE we implement our technique on ISCAS'85 and ITC'99 benchmarks. We used two small (C531, C7552), two medium (b14, b15) and two big (b21, b22) circuits. All the attack codes have been compiled and run on a single node with Intel processor running at 4GHz and 12 GB of RAM with UBUNTU 20.04.3. For reporting area, power, and delay (APD) overheads, we use Cadence Genus along with 45nm NanGate Open Cell Library.

## 4.2 Security Analysis

**Class A**: To investigate the effectiveness of the ENTANGLE against the FALL [11] and SFLL-HD-Unlocked [13] attacks we secured the benchmarks using $ENTANGLE - HD^{k=64}$. Both attacks were able to find all the members of the protected input set successfully; however, both algorithms terminate in less than 20 seconds because they could not identify any potential candidates as (*PP*) signal. The reason lies in the fact that ENTANGLE breaks the single signal (*PP*) that contains all the relevant combinations of the protected pattern. As a result, these algorithms fail in finding a single signal with intended properties rendering the ENTANGLE resilient against class A attacks.



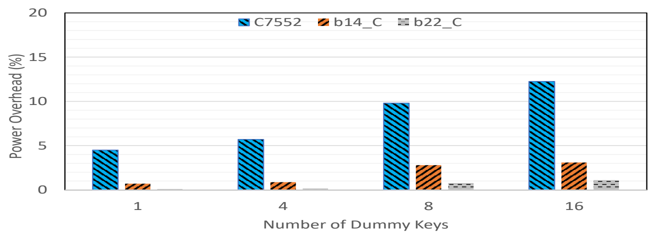


**Figure 6: Area and power overhead for dummy key bit technique for ENTANGLE with k=64**

**Class B**: For ML-based attacks, we added a dummy logic unit to the C7552 benchmark using the same training set as [1]. We observed that by increasing the number of arbitrary nodes incorporated in forming the *DS* signal, the Macro F1-Score starts to decrease for $ENTANGLE - HD^{k=64}$. We then added dummy key units to the restore unit with the same PIs that are in the transitive fan-in cone of the *DS* signal. By following the *DS* signal, we infer that the post-processing algorithm misclassifies at least 36% of the perturb nodes as design nodes, among them *PP1* and *PP2*.

## 4.3 Overhead Analysis

**ENTANGLE** : Table 1 shows the area, power, and delay overhead of $ENTANGLE - HD^{k=64}$ compared with the original design for *k=64*. As can be seen, the overhead of ENTANGLE is high for smaller circuits because of adding k-bit comparator and adder. These overheads become less in larger benchmarks showing that the ENTANGLE fits well in large-sized industrial circuits. Note these overheads are negligible comparing SFLL-HD [7]. The delay overhead shown in Table 1 is the overhead in the critical path delay. As the circuit becomes larger, the synthesis tools implement the design differently, considering the area optimization. So, the critical path may change, or the delay of this path may decrease. That is the reason for the negative numbers in the delay report.

**Dummy key**: For calculating the area, delay, and power overheads of the *Dummy key* technique, we locked c7552, b14, and b22 circuits using ENTANGLE and added 1, 4, 8 ,and 16 dummy key bits. In our implementation, we AND each dummy key bit with a PI from the unprotected set. Figure 6 shows the area and power overheads for different numbers of dummy keys in the ENTANGLE . Again, the overhead is acceptable for the larger benchmarks. We have not reported the delay overhead because the critical path of the circuit remained unchanged for all the implementations.

**Dummy logic**: The area and power overheads of the dummy logic are mainly related to the *h+1* comparator and new 1-bit full adders, which is negligible, especially for larger circuits. However, the delay overhead of this technique on the protected output is directly related to the *DS* signal. We implemented the dummy logic on b22_C benchmark. The area and power overheads are 1% and 0.04%, respectively. Adding dummy logic did not change the delay of the circuit in our implementation.

## 5 CONCLUSION

In this paper, we proposed ENTANGLE , a new stripped-functionality logic locking schema that can protect the circuit against the SAT, approximate, and the state-of-the-art structural attacks designated for SFLL. We show that breaking the tip of the perturb unit and blending it with the design output fails the algorithmic attacks in finding this signal and inferring the correct key from it. For further fortifying the ENTANGLE against the ML-based attacks, we introduced two cloaking techniques to misguide the neural network in the classification stage. we showed that ENTANGLE can secure the large-sized industrial circuits with an average overhead of 11.6 percent and 9.1 percent for area and power, respectively.

**Table 1: Area, power, and delay overheads for ENTANGLE with k=64. Negative values indicate improvement in specific values**

| Benchmark | Area (%) | Power (%) | Delay (%) |
|---|---|---|---|
| C5315 | 48.36 | 56.71 | 5.6 |
| C7552 | 47.26 | 55.23 | 6.3 |
| b14_C | 29.39 | 27.86 | -2.9 |
| b15_C | 18.43 | 17.35 | -1.8 |
| b21_C | 14.51 | 11.07 | 0.6 |
| b22_C | 8.82 | 7.14 | -4.3 |

## 6 ACKNOWLEDGMENT

This work was financially supported by the German Federal Ministry of Education and Research BMBF under the framework of VE-CirroStrato.

## REFERENCES

[1] Lilas Alrahis et al. 2021. GNNUnlock: Graph neural networks-based oracle-less unlocking scheme for provably secure logic locking. In *DATE*. IEEE.
[2] Meng Li et al. 2017. Provably secure camouflaging strategy for IC protection. *IEEE TCAS* 38 (2017).
[3] Jeyavijayan Rajendran et al. 2012. Security analysis of logic obfuscation. In *DAC*.
[4] Jarrod A Roy et al. 2008. Markov. EPIC: Ending piracy of integrated circuits. In *DATE*.
[5] Akashdeep Saha et al. 2020. Lopher: Sat-hardened logic embedding on block ciphers. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.
[6] Abhrajit Sengupta et al. 2018. ATPG-based cost-effective, secure logic locking. In *IEEE VTS*. IEEE.
[7] Abhrajit Sengupta et al. 2020. Truly stripping functionality for logic locking: A fault-based perspective. *IEEE TCAD* 39 (2020).
[8] Kaveh Shamsi et al. 2017. AppSAT: Approximately deobfuscating integrated circuits. In *HOST*.
[9] Kaveh Shamsi et al. 2018. On the approximation resiliency of logic locking and IC camouflaging schemes. *IEEE TIFS* 14 (2018).
[10] Yuanqi Shen and Hai Zhou. 2017. Double DIP: Re-evaluating security of logic encryption algorithms. In *GLSVLSI*.
[11] Deepak Sirone and Pramod Subramanyan. 2020. Functional analysis attacks on logic locking. *IEEE TIFS* 15 (2020).
[12] Pramod Subramanyan et al. 2015. Evaluating the security of logic encryption algorithms. In *HOST*. IEEE.
[13] Fangfei Yang et al. 2019. Stripped functionality logic locking with Hamming distance-based restore unit (SFLL-hd)–unlocked. *IEEE TIFS* 14 (2019).
[14] Muhammad Yasin et al. 2015. On improving the security of logic locking. *IEEE TCAD* 35 (2015).
[15] Muhammad Yasin et al. 2016. SARLock: SAT attack resistant logic locking. In *HOST*.
[16] Muhammad Yasin et al. 2017. Provably-secure logic locking: From theory to practice. In *CCS*.
[17] Muhammad Yasin et al. 2017. Removal attacks on logic locking and camouflaging techniques. *IEEE TETC* 8 (2017).