InterLock: An Intercorrelated Logic and Routing Locking

Hadi Mardani Kamali[1], Kimia Zamiri Azar[1], Houman Homayoun[2], Avesta Sasan[1]
[1] George Mason University, Fairfax, VA, USA.
{hmardani,kzamiria,asasan}@gmu.edu
[2] University of California, Davis, Davis, CA, USA.
{hhomayoun}@ucdavis.edu

ABSTRACT

In this paper, we propose a *canonical prune-and-SAT* (*CP&SAT*) attack for breaking state-of-the-art routing-based obfuscation techniques. In the *CP&SAT* attack, we first encode the key-programmable routing blocks (keyRBs) based on an efficient SAT encoding mechanism suited for detailed routing constraints, and then efficiently re-encode and reduce the CNF corresponded to the keyRB using a bounded variable addition (BVA) algorithm. In the *CP&SAT* attack, this is done before subjecting the circuit to the SAT attack. We illustrate that this encoding and BVA-based pre-processing significantly reduces the size of the CNF corresponded to the routing-based obfuscated circuit, in the result of which we observe 100% success rate for breaking prior art routing-based obfuscation techniques. Further, we propose a new *intercorrelated logic and routing locking* technique, or in short *InterLock*, as a countermeasure to mitigate the CP&SAT attack. In Interlock, in addition to hiding the connectivity, a part of the logic (gates) in the selected timing paths are also implemented in the keyRB(s). We illustrate that when the logic gates are twisted with keyRBs, the BVA could not provide any advantage as a pre-processing step. Our experimental results show that, by using InterLock, with only three 8×8 or only two 16×16 keyRBs (twisted with actual logic gates), the resilience against existing attacks as well as our new proposed *CP&SAT* attack would be guaranteed while, on average, the delay/area overhead is less than 10% for even medium-size benchmark circuits.

KEYWORDS

Logic Obfuscation, Routing Obfuscation, The SAT Attack

ACM Reference Format:
Hadi M Kamali, Kimia Z Azar, Houman Homayoun, and Avesta Sasan. 2020. InterLock: An Intercorrelated Logic and Routing Locking. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3400302.3415667*

1 INTRODUCTION

The globalization of the design and implementation of integrated circuits has drastically increased, particularly in the past two decades. This is when high-tech companies try (1) to reduce the cost of manufacturing, (2) to access technology that is inclusively available by a limited number of suppliers, (3) to reduce time to market, and (4) to meet the market demand [6]. However, it has also raised many security threats and trust challenges. Some of these threats include that of IC overproduction, Hardware Trojan insertion, reverse engineering, and Intellectual Property (IP) theft [44].

To combat these threats, numerous *Design-for-Trust* (*DfTr*) techniques have been proposed, one of them is *Logic locking* [24, 26], *a.k.a* logic obfuscation. In Logic locking, the designer adds post-manufacturing programmability into the design controlled by programmable values referred to as the *key*. The key value is driven from an on-chip tamper-proof non-volatile memory (tpNVM) [57], and it will be initiated after fabrication via a trusted party. Hence, the adversary cannot recover the correct functionality of a logic locked chip without having the correct key.

The security and the strength of the primitive logic locking techniques [24–26] has been called into question by various attacks, especially the *Boolean satisfiability* (SAT) based attack [40, 56]. In the SAT attack, it is assumed an adversary has access to (1) an oracle (working chip), (2) a fully reverse engineered netlist, and (3) the scan chain access of oracle (writing/reading the content of internal registers at will.). Based on these assumptions, the SAT attack, as an *oracle-guided* attack, starts iteratively ruling out the set of incorrect keys using a few selected input queries found by the SAT solver, called discriminating inputs (DIPs) [40, 56].

To thwart the SAT attack, over the past few years, researchers have investigated four different categories [36]:

(1) **Point Function Based Obfuscation:** The first group of techniques, examples of which include SARLock, Anti-SAT, and SFLL [47, 48, 69], tries to reduce the strength of the SAT attack such that each DIP could only rule out one incorrect key (or a few). So, it significantly increases the number of required SAT iterations (required DIPs). However, they are vulnerable against structural-based attacks [11, 49, 50, 66]. Besides, these techniques suffer from very low output corruption, making them susceptible to approximate-based attacks [28, 67].

(2) **Behavioral/Cyclic Obfuscation:** In the second group of techniques, such as delay locking [70], timing-based locking [16, 38], or cyclic locking [4, 5, 29, 61, 62], the obfuscated circuit (I) is not translatable to a SAT problem (delay/timing based), or (II) traps the SAT solver in an infinite loop (cyclic), or (III) it leads to an incorrect key (cyclic). However, the existing techniques in this category are already broken using SMT attack (delay) [34], timingSAT attack (timing) [3], and the SAT-based attacks on cyclic locking [21, 30, 68].

(3) **Scan Chain Blocking/Obfuscation:** Since many of prevailing attacks rely on access to the scan chain, the third group of techniques locks/blocks the scan for any unauthorized scan access [18, 19, 33, 53, 59, 63, 65]. Since the SAT attack is only applicable to combinational circuits, when the scan chain is blocked/obfuscated, the adversary's access is limited to only primary inputs/outputs

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-6654-2324-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415667

(PI/PO). Hence, the SAT attack is no longer applicable to the whole (sequential) circuit. However, these techniques are later broken using unrolling-based SAT attacks as well as the SAT attack integrated with BMC [31, 37, 41]. Also, blocking the scan chain enforces the tester to rely on the PO for any test/debug purpose, which might reduce the test coverage considerably.

(4) **Symmetric Interconnection Obfuscation:** Taking a step further, the fourth group of techniques tries to significantly increase the complexity of inner calculations of the SAT solver leading to an extremely long runtime per *each iteration* of the SAT attack [17, 32, 60]. These techniques rely on building symmetric interconnection into the locked portion of the circuit, extremely increasing the depth of the SAT search tree, and reducing the number of derived variables (implications) based on assigned variables. The building block of the existing solutions in this category are the key-programmable routing blocks (we call them *keyRB* in this paper), each has its topology, such as crossbar or permutation (logarithmic) network. Although this group of obfuscation techniques suffers from the higher area/delay overhead, to the best of our knowledge, there is still no attack on this category of logic locking techniques.

1.1 Contribution

In this paper, we first propose a *canonical prune-and-SAT* (*CP&SAT*) attack on the fourth group of techniques (The first attack on routing-based obfuscation). In our proposed *CP&SAT* attack, we first extract and model the circuit into some numerical bound problems, which could be re-encoded efficiently using a bounded variable addition (BVA) algorithm. Then, the BVA algorithm is applied to each numerical bound problem, separately. The BVA re-encodes and reduces the CNF size/complexity of each numerical bound problem significantly. After reduction using the BVA, the reduced CNFs (corresponded to numerical bound problems) will be merged again with the circuit's CNF, and the SAT solver could be executed on the reduced CNF version. Our security analysis on benchmark circuits protected by existing routing obfuscation techniques, such as Cross-Lock and Full-Lock [17, 32], demonstrates 100% successfulness of this attack within a short time.

We then propose an enhanced obfuscated technique, that mitigates the weakness of existing routing-based obfuscation techniques [17, 32] against the proposed *CP&SAT* attack. We refer to our proposed obfuscation solutions as *InterLock*. In *InterLock*, the routing obfuscation (keyRB) is *intercorrelated* with logic obfuscation. Hence, since the logic is truly twisted with routing all controlled by the key, it is not possible to convert and model the keyRB(s) into the numerical bound problem(s), and consequently, the BVA is no longer applicable to them for reduction.

We implement and evaluate the keyRBs in *InterLock* based on three different technologies: (1) transmission-gate (Tgate) CMOS, (2) programmable-via using anti-fuse elements (PVIA), and (3) three-independent-gate field-effect transistors (TIGFET). It helps us to provide a better illustration of the area/delay overhead. We also show that by implementing in the lower level of abstraction, the area/delay overhead of *InterLock* could be even below ~10% to make the design resilient against the prevailing attacks.

2 BACKGROUND

2.1 The Oracle-guided SAT Attacks

In the SAT attack, with having access to (1) a working chip with open scan chain, and (2) the reverse-engineered netlist, each combinational part of the circuit could be evaluated independently. For any arbitrary combinational part (which is locked), $c_{lock}: I \times K \rightarrow O$, where $K = \{0,1\}^k$ is the *key* space, there exists $k_c \in K$ such that $\forall i \in I \Rightarrow c_{lock}(i, k_c) = c_{oracle}(i)$, and c_{oracle} is the combinational logic part of working chip (*oracle*).

In the SAT attack, by getting inspiration from the miter circuit used in formal verification (equivalency checking), a miter circuit has been built as $miter \equiv c_{comb\_lock}(dip, k_1) \neq c_{comb\_lock}(dip, k_2)$. This miter circuit returns a specific *discriminating input pattern* (dip) that produces different output using two different keys k_1 and k_2. Then, this dip is queried on the oracle $eval \leftarrow c_{comb}(dip)$ and a new I/O constraint will be generated. This new I/O constraint $c_{comb\_lock}(dip, k_1) = c_{comb\_lock}(dip, k_2) = eval$ is stored back in the solver and the $miter$ circuit would be solved again to find a new dip. When the $miter$+constraints problem has no longer satisfying assignment (no new dip), the constraints could identify a $k_c \in K$.

2.2 Point Function Based Obfuscation

Based on the iterative structure of the SAT attack, its runtime could be obtained from:

$$T_{Attack} = \sum_{i=1}^{N} T(i) = \sum_{i=1}^{N} (t_i + T_{DPLL}(\Phi_i)) \quad (1)$$

In Eq. 1, the T_{DPLL} is the runtime of the DPLL (*Davis-Putnam-Logemann-Loveland*) algorithm, t_i is the runtime of the remaining book-keeping code executed at each iteration i, and N is the number of iterations (number of DIPs) required for de-obfuscation. The DPLL (or one of its derivatives) is a recursive algorithm that used to perform *Conflict-Driven Clause Learning* (*CDCL*) as the main part of the SAT solver. Getting the benefit of this recursive algorithm, the SAT attack can recover the correct functionality with fast convergence.

The most intuitive mechanism to combat against the srength of the SAT attack is to find a way to maximize N, which is the main aim of point function based obfuscation techniques (category 1). [47, 48, 69]. For this purpose, the structures proposed in this category weakens the pruning power of each dip, guaranteeing that each dip can only rule out one (or a small number of) incorrect key(s). This forces the number of needed iterations (N) to exponentially increase with respect to the number of keys. However, with a very shallow DPLL recursive tree, the execution time of each iteration of the SAT solver ($t + T_{DPLL}$) is quite short.

2.3 Symmetric Interconnection Obfuscation: One Step Deeper

Symmetric interconnection obfuscation techniques show how the direction of adding difficulties to the SAT attack could be changed (becoming deeper) via deepening the DPLL tree. This deepening could be achieved if the obfuscation circuit forces a relationship between the number of clauses and the number of variables to maximize the penalty associated with incorrect variable assignment symmetrically across the search tree. With having a deeper DPLL tree, the runtime formula of the SAT attack (Equation 1) could be re-written as follows [17]:

$$T_{Attack} = \sum_{i=1}^{N} (t_i + T_{DPLL}(\Phi_i)) \simeq \sum_{i=1}^{N} \sum_{j=1}^{M} (T_{DPLL}^{Avg}) \quad (2)$$

The main aim of symmetric interconnection obfuscation techniques (category 4) is to extremely increase the number (M) and the computational complexity (T_{DPLL}^{Avg}) of recursive calls in DPLL algorithm,

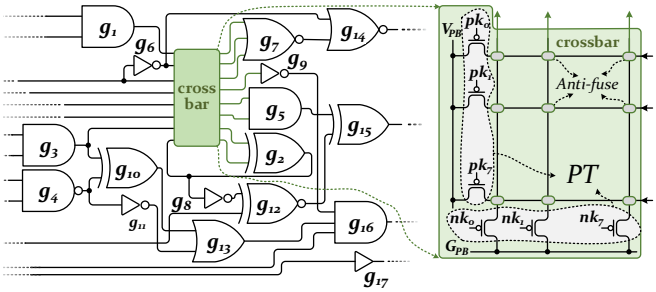


Figure 1: Circuit Locked by Cross-Lock [32] with an 8×8 Crossbar Network.

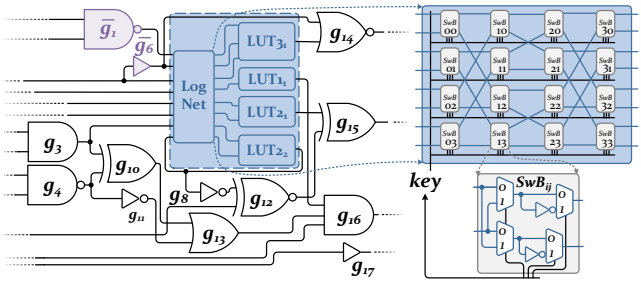


Figure 2: Circuit Locked by Full-Lock [17] with an 8×8 Logarithmic Network.

which occurs when the DPLL tree is extremely deep/large enough. Two examples of this category are Cross-Lock [32] and Full-Lock [17].

2.3.1 *Cross-Lock: PVIA-based Crossbar Obfuscation*

In Cross-Lock [32], each keyRB are built using programmable-vias (PVIA) constructed using one-time-programmable (OTP) elements made of anti-fuse [7]. PIVAs are used to implement $n \times m$ crossbars. The main approach for PVIA programming is connecting two complementary (NMOS and PMOS) programming transistors (PTs) to the two ends that connect the device terminals to programming supplies [7]. Routing obfuscation in Cross-lock has been performed by inserting PVIA-based $n \times m$ crossbars. Fig. 1 shows a small circuit locked by Cross-Lock with $n = m = 8$.

2.3.2 *Full-Lock: Logarithmic-based Logic+Routing Obfuscation*

In Full-Lock [17], the keyRB(s) is built using logarithmic near non-blocking routing network. Since the SAT solver accepts the inputs in conjunctive normal form (CNF), Full-Lock was motivated by [10], where it was shown that the number of recursive calls in the DPLL algorithm could be maximized when the *clause to variable ratio* of a CNF is close to 4. Then, a symmetric logarithmic routing network is introduced in Full-Lock that forces this ratio to 4. Also, to elevate the complexity of the obfuscated circuit against the SAT attack, the logic gates succeeding each keyRB are replaced with same-size look-up-tables (LUT). Fig. 2 shows a small circuit (similar circuit with Fig. 1) that is locked by Full-Lock with $n = m = 8$, and the succeeding gates (g_2, g_5, g_7, and g_9) are replaced with same-size LUTs ($LUT_{2(1)}$, $LUT_{2(2)}$, $LUT_{3(1)}$, and $LUT_{1(1)}$, respectively). Also, in each key-programmable switch-box (*SwB*), a key-programmable inverter has been added, allowing output to be negated based on the key value. This negation capability allows us to replace the logic gates preceding each keyRB with their negated version (g_1: *AND* $\rightarrow$ *NAND*, g_6: *NOT* $\rightarrow$ *BUFF*), and handle the negation within the keyRB.

2.4 Cases Requiring Cyclic-based SAT Attack

By using a keyRB for routing-based obfuscation, when the selected nets are correlated, an incorrect key might generate a feedback, which results in the generation of a combinational cycle. For example, when a net is in the fan-in-cone of another net, and both are selected as input to the keyRB, an incorrect key most likely generates a combinational cycle (e.g. $g_2 \rightarrow$ *crossbar* $\rightarrow g_2$ in Fig. 1).

Since the SAT attack is only applicable to *directed acyclic graphs* (DAG), the generation of cycles mislead (to an incorrect key, or an infinite loop) the SAT attack. The existence of cycles, however, does not prevent the SAT attack formulation. In many studies on cyclic obfuscation [21, 30, 68], it was shown that by adding a pre-processing step to the SAT attack, it could add necessary cycle avoidance clauses for a successful SAT attack in the presence of combinational cycles. From this argument, for the security analysis of routing obfuscation, a cyclic-based SAT attack must be used.

3 CANONICAL PRUNE-AND-SAT ATTACK

As of today, there is still no successful attack on routing-based obfuscation. Each iteration of SAT solving on routing-based obfuscated circuits faces an ultra-deep and complex DPLL tree. Hence, the SAT attack cannot even find a satisfying assignment(s) to complete the de-obfuscation process. However, in this work, we propose *canonical prune-and-SAT* (*CP&SAT*) attack on the routing-based obfuscation techniques. In the *CP&SAT*, we first model the key-programmable routing blocks (keyRB(s)) as numerical bound problems, and then a bounded variable addition (BVA) algorithm has been engaged as a pre-processing step to reduce the size and complexity of numerical bound problems. By using the BVA algorithm, the CNF corresponded to each keyRB will be reduced dramatically in terms of the number of clauses. Then, the re-encoded CNF will be solved using the traditional SAT attack.

3.1 Threat Model in *CP&SAT* Attack

The *CP&SAT* attack will be performed based on the conventional threat model for logic locking [24, 40, 56], where:

(1) The adversary has access to the successfully reverse-engineered yet locked netlist. Hence, (s)he has all the necessary information about the netlist, such as the obfuscation technique, the key gates, the key inputs, etc. Specifically in routing obfuscation, the location of the keyRBs could be determined by the adversary.
(2) The adversary has access to an activated/unlocked chip, in which the correct key is embedded into a secure tpNVM.
(3) With having scan chain access on the activated/unlocked chip, the adversary can apply the SAT attack on each combinational part of the circuit, independently.

3.2 Attack Flow

The proposed *CP&SAT* attack is composed of three main steps: (1) modeling the keyRB(s) to be presented as a numerical bound problem, where for each output of keyRB, a sub-CNF will be extracted from the CNF of the whole circuit; (2) re-encoding the sub-CNF corresponded to each keyRB output using bounded variable addition (BVA) algorithm; (3) merging the updated (reduced) sub-CNFs into the CNF of the whole circuit, running the traditional SAT attack, and match the key for the correct routing.

3.2.1 *Modeling keyRB as a Numerical Bound Problem*

Extensive analysis on the application of Boolean satisfiability in detailed routing constraints [8, 14, 15, 45, 46] shows that the SAT

solvers can consider simultaneously the routability constraints for all nets, leading to potentially faster convergence to a solution. However, this only happens when an appropriate *encoding approach* has been chosen to represent routing constraints as a SAT problem before solving. Many studies have investigated and compared different encoding approaches [15, 45]. Using the key observations provided in these studies, in the first step of the proposed *CP&SAT* attack, we encode the sub-CNF related to each keyRB output using *one-layer linear encoding*. To describe the logic-equivalent model, for each output of a $n \times n$ keyRB, the one-layer linear encoding replaces the original sub-CNF with a CNF describing a one-layer $n{-}to{-}1$ multiplexer (MUX) controlled by the one-hot key. More formally, for a $n \times n$ keyRB, the sub-CNF of each keyRB output, which is encoded using one-layer linear encoding, will be as follows:

$$\bigwedge_{\substack{M\subseteq 1,\ldots,n,\\ |M|=1}} \Big(\bigvee_{i\in M} x_i k_i\Big) \tag{3}$$

In which x_i denotes the wire that is connected to the i^{th} input of the keyRB, and k_i denotes the one-hot key that connects the i^{th} input of the keyRB to the corresponded keyRB output when it is 1, and M is the search space for each keyRB output. The Eq. 3 is the most special case of encoding of numerical bounds [54]. The numerical bound problems could be denoted as $\leq p(x_1, x_2, ..., x_n)$, meaning that among n variables p variables are allowed to be assigned true. The most special case of numerical bounds is when $p = 1$, called *at-most-1 constraint*, that is applied whenever a finite domain is encoded, and the Eq. 3 is one form of at-most-1 constraint encoding. According to this encoding definition, in the first step of the proposed *CP&SAT* attack, we first extract the sub-CNF related to each output of keyRB(s). Then, we use *one-layer linear encoding* for the extracted sub-CNF to be encoded as a numerical bound problem. Then, in the second step as described in the next section, we use the BVA to re-encode and reduce the size of each sub-CNF for each output of the keyRB.

3.2.2 SAT Reduction using Bounded Variable Addition

As an integral part of SAT solving, *resolution* and *variable elimination* (VE) are two rules that would be applied on CNF before running the SAT solver to reduce the size of variables/literals [1, 39, 52]. The VE, as a proof procedure for CNF formulas, faces an exponential space complexity. Hence, to make it practical for usage, the VE must be bounded [1, 52]. In bounded VE (BVE) a variable x could be eliminated only if $|S| \leq |S_x \cup S_{\bar{x}}|$, in which $S_x(S_{\bar{x}})$ denotes a set containing clauses all contain $x(\bar{x})$, S is obtained from Eq. 4, and $|S| \leq |S_x \cup S_{\bar{x}}|$ means that the resulting CNF $\big(((F \setminus (S_x \cup S_{\bar{x}})) \cup S\big)$[1] will contain no more than the original CNF (F) clauses.

$$S = S_x \otimes S_{\bar{x}} = \{C_1 \otimes C_2 | C_1 \in S_x, C_2 \in S_{\bar{x}}, C_1 \otimes C_2 \neq Tautology\} \tag{4}$$

In *CP&SAT attack*, we engage the complementary version of BVE, called *bounded variable addition* (BVA) [54], in which either a new variable will be added to the CNF or a variable will be substituted. Similar to BVE, the same *bounding* concept must be used in BVA to decrease the size of the CNF [54]. As the simplest example of the BVA, by adding a new variable x to the following formula F with 6 clauses, the re-encoded formula F' would have one clause less.

$$F = (a \vee c) \wedge (a \vee d) \wedge (a \vee e) \wedge (b \vee c) \wedge (b \vee d) \wedge (b \vee e) \tag{5}$$

[1] $((F \setminus (S_x \cup S_{\bar{x}})) \cup S$ means that, in CNF F, both sets of clauses, S_x and $S_{\bar{x}}$, that contain x and $\bar{x}$, respectively, must be replaced with clauses of set S that are built using Eq. 4.

$$F' = (a \vee x) \wedge (b \vee x) \wedge (c \vee \bar{x}) \wedge (d \vee \bar{x}) \wedge (e \vee \bar{x}) \tag{6}$$

In the BVA, the number of possibilities to add or substitute a variable is extremely large. Hence, to make it practical for any CNF (F), the BVA algorithm must be constructed based on two steps:

(1) *Replaceable Matching*: Creating a pair of sets consisting of a set of literals (SET_L) and a set of clauses (SET_C) such that for all $\{l, c\} \in \{SET_L, SET_C\}$, the clauses $(c \setminus \{SET_L\}) \cup \{l\}$ are either in CNF (F) or tautological.
(2) *matching-to-clauses*: Using a method that creates the sets $S_x = \{(l \vee x) \mid l \in SET_L\}$ and $S_{\bar{x}} = \{(c \setminus SET_L) \cup \{\bar{x}\} \mid c \in SET_C\}$, and removes all clauses $(c \setminus \{SET_L\}) \cup \{l\}$, and replaces them with $S_x \cup S_{\bar{x}}$.

By applying these two steps on F (Eq. 5), $SET_L = \{a, b\}$ and $SET_C = \{(a \vee c), (a \vee d), (a \vee e)\}$, F' could be generated using *matching-to-clauses* with one clause less as shown in Eq. 6. Now, by using this 2-step BVA algorithm, any CNF formula could be reduced provably in size (the number) of clauses while the reduced CNF is also provably equivalent with the original CNF.

Theorem 1. *For two sets as* replaceable matching $\{SET_L, SET_C\}$ *as for the CNF formula F, F' as the reduced of F could be constructed by adding a Boolean variable such that (1) F' is logically equivalent to F and (2) F' contains $|F'| + |SET_C| + |SET_L| - |SET_C| \times |SET_L|$ clauses if none of the resolvents is a tautology.*

Proof. For two sets as replaceable matching $\{SET_L, SET_C\}$, we can construct F' as follows: All clauses of $(c \setminus \{SET_L\}) \cup \{l\}$ must be removed from F and must be replaced with $S_x \cup S_{\bar{x}}$ that are obtained using the matching-to-clauses construction method. The number of removed clauses is $|SET_L| \times |SET_C|$, while the number of added clauses is $|SET_L| + |SET_C|$ proving (2). Since the BVA is the complement of BVE, by applying BVE on x in F', it re-produces (reverse) F, and BVE preserves logical equivalence proving (1). ■

One of the best fitting applications of the BVA algorithm is re-encoding cardinality constraints [2, 8, 14, 54], where it is necessary to encode numerical bounds ($\leq k(x_1, x_2, ..., x_p)$). As discussed previously, the one-layer linear encoding formulates each keyRB output as the most special case of cardinality constraints ($k = 1$), called *at-most-1 constraint*. Compared to naive encoding for *at-most-1 constraint* in which the number of clauses is $n.(n+1)/2$, by using BVA algorithm, the number of clauses would be reduced to $\sim 3n$ [54].

It is worth mentioning that numerous studies are explaining how cardinality constraints (numerical bounds) could be encoded efficiently [23, 43, 55, 58, 64]. Also, a few SAT solvers handle cardinality constraints by itself, such as Sat4J [9] or clasp [42]; however, since these solvers do not extract cardinality constraints from the formula, compared to the direct re-encoding using BVA, their efficiency is extremely low. Furthermore, the strongest SAT solvers tend to not support native cardinality constraints, such as MiniSAT [51] that was supporting native cardinality constraints up to version 1.12. In *CP&SAT attack*, we employ the simpleBVA proposed in [54] as a pre-processing step before running the SAT attack and after one-layer linear encoding. The simpleBVA will be used for each sub-CNF, corresponded to each keyRB output, and encoded using one-layer linear encoding, separately.

3.2.3 SAT Execution and Key Matching

After the reduction using the BVA algorithm, we update the CNF of the whole circuit using the reduced sub-CNFs corresponded to keyRB(s) outputs. Now, it is time to run the traditional SAT attack on the updated CNF. Since each keyRB might add cycles into the

design, as mentioned in Section 3.1, we need to use a cyclic-based SAT attack. Assuming that the circuits are acyclic, the CycSAT-I[2] will be used.

As was mentioned previously, in one-layer linear encoding, the actual keys of each keyRB will be replaced with a set of one-hot key controlling the MUXes (one-layer encoding). Hence, after de-obfuscating the updated CNF, the SAT attack will recover the values of the one-hot keys. These one-hot keys determine the correct wiring/interconnection for the MUXes. So, a matching step is required, in which we need to calculate the actual key for each keyRB that establishes the same (correct) wiring/interconnection built by one-hot key in MUXes.

4 INTERLOCK: RESISTING CP&SAT ATTACK

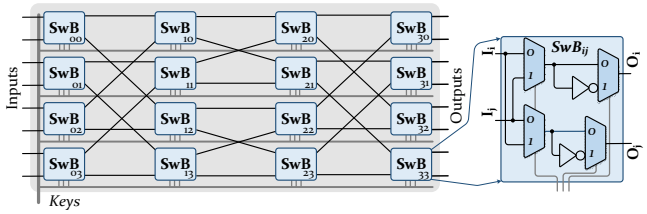
In the previous section, we described how prior routing-based obfuscation techniques could be broken using the proposed *CP&SAT*. It calls into a question that *"How routing obfuscation could be still used while it is not vulnerable to the BVA algorithm?"*. In this section, we answer this question by proposing a countermeasure that improves the resiliency of this category of obfuscation techniques against *CP&SAT* attack.

4.1 Truly-Twisted Logic & Routing Obfuscation

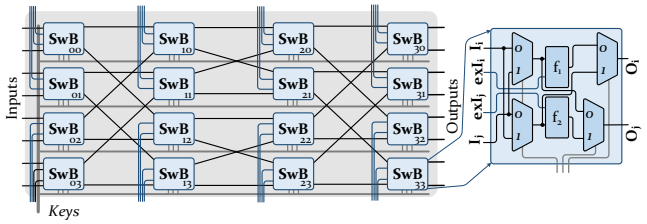
To still get the benefit of routing obfuscation, and to combat against the efficiency of BVA-based re-encoding on routing-based obfuscation, we truly twist the keyRB with logic gates, meaning that a part of the actual logic gates will be also embedded into the keyRB.

In the *CP&SAT* attack, we explained how a keyRB could be modeled as multiple numerical bound problems before the BVA re-encoding. So, the idea is that when the routing-based obfuscated circuit could not be translated (converted) to a numerical bound problem, the BVA is no longer applicable to it. For this purpose, inspired by the logarithmic (permutation) networks proposed in Full-Lock [17], we employ the same architecture for keyRBs in InterLock; however, for each layer of that hierarchy, we will add a Boolean function (logic gate). Fig. 3 shows our new keyRB architecture that must be used for routing-based obfuscation. Compared

[2]CycSAT-I is designed and applicable to cyclic obfuscation when the original circuit is acyclic.



(a) key-programmable Routing Block (KeyRB) in Full-Lock [17]



(b) key-programmable Routing Block (KeyRB) in our Proposed InterLock

Figure 3: Full-Lock [17] vs. Our Proposed InterLock.

Table 1: The SAT Attack Runtime on ISCAS-85 c7552 with only One KeyRB in Different Scenarios.

$f_{1,2}$ / Size	Full-Lock [17]	InterLock all *NAND*	InterLock all *NOR*	InterLock all *XNOR*	InterLock Random
keyRB-4	0.02	0.192	0.136	0.718	0.232
keyRB-8	0.437	3.083	5.905	2062	19.79
keyRB-16	5.413	522.1	558.2	Timeout	62332
keyRB-32	195.1	Timeout	Timeout	Timeout	Timeout
keyRB-64	Timeout	Timeout	Timeout	Timeout	Timeout

Timeout = 10^5 *Seconds* ≈ *1 day*

to Full-Lock [17], for each switch-box (SwB), the configurable inverters are removed, and f_1 and f_2 are added that could be any of 2-input basic logic gates, i.e. *NAND, NOR, XNOR, AND, OR, XOR*. Also, for each SwB, we add extra inputs (*exI*) as one of the inputs of 2-input logic gates. For each SwB with 4 inputs (I_i, I_j, exI_i, exI_j), output O_i could be $\{I_i, I_j, f_1(I_i, exI_i)$, and $f_1(I_j, exI_i)\}$, and output O_j could be $\{I_i, I_j, f_2(I_i, exI_j)$, and $f_2(I_j, exI_j)\}$.

4.1.1 *Different Possibilities for f_1 and f_2*

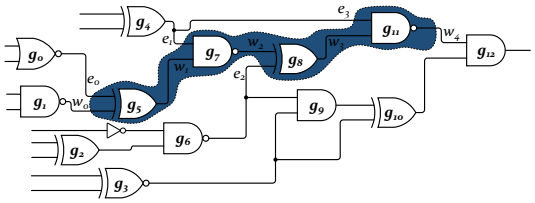
In this section, we aim to explain (1) *"Why the usage of f_1 and f_2 gates in the SwBs improves the resiliency of Interlock (compared to full-lock in which only routing and inversion is implemented)?"*, and (2) *"how the selection of the logic for f_1 and f_2 affects its resiliency"*. To answer these two questions, we investigate five different scenarios: f_1s and f_2s could be (1) still inverters with no extra input (similar to Full-Lock), (2) all *NAND* (*AND*), (3) all *NOR* (*OR*), (4) all *XNOR* (*XOR*), and (5) selected randomly (any arbitrary 2-input gate).

Table 1 shows the runtime of the SAT attack (CycSAT-I) on an obfuscated ISCAS-85 c7552 circuit while only one keyRB is embedded into the design based on these five scenarios. The first and the most promising observation is that, compared to Full-Lock (when the logic layer is still inverter), for the same-size keyRB, the InterLock (all scenarios) builds a much harder SAT problem. As shown, in Full-Lock, the smallest single keyRB that breaks the SAT attack is a 64×64 keyRB (keyRB-64). However, in InterLock, it is even smaller (keyRB-16 or keyRB-32). Furthermore, we observe that, while all f_1s and f_2s are *XNOR* (or *XOR*), keyRB-16 is enough; however, for other gate types (*NAND, NOR, AND, OR*), the smallest resilient is keyRB-32.

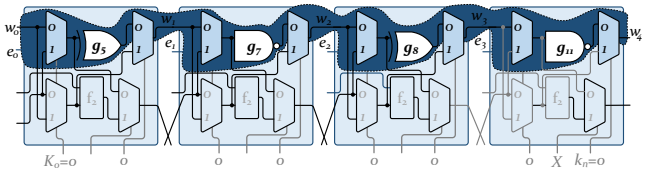
4.1.2 *Embedding Actual Timing Paths into KeyRBs*

This is a key observation that when all f_1s and f_2s are *XNOR* (*XOR*), the SAT resiliency of the keyRB is extremely higher. But, as shown in Fig. 3, similar to Full-lock that engaged inverters to handle the toggling of some gates preceding the keyRB, in InterLock, these extra gates must become a part of the actual logic gates to avoid far exceeding the overhead. However, it is less likely to find a set of paths that only consist of *XNOR* (*XOR*). Hence, if we select the gate of each SwB based on an actual gate in a selected timing path, all f_1s and f_2s will become the actual gates of the design. It guarantees that, in InterLock, only MUXes could be considered as the overhead, however, in Full-Lock, all inversions except one layer are surplus as an extra overhead. Hence, although InterLock adds extra logic, it even reduces the overhead compared to the Full-Lock.

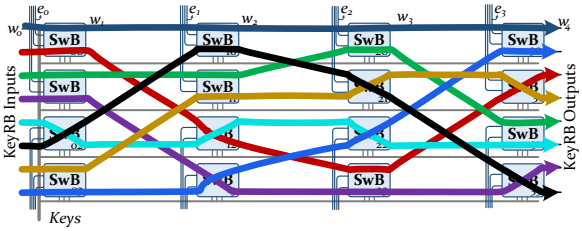
To embed part(s) of the logic gates into each keyRB, we need a strategy to select the gates from the design. For the selection strategy, when the number of timing paths in each keyRB is m, m



(a) A Timing Path Selected to be Embedded into KeyRB



(b) Inserting the Selected Timing Path into a KeyRB



(c) 100% Utilization: m Timing Paths in a $m \times m$ KeyRB

Figure 4: Timing Path Embedding into KeyRB.

timing paths must be selected[3] For $2log_2(m) - 2$ layers of SwBs in permutation-based network [17][4], the length of the timing path must be equal with $2log_2(m)-2$. Hence, among the candidate timing paths, we select paths (or cut the paths) with length $2log_2(m)-2$. For example, Fig. 4 shows how an actual timing path will be embedded into a keyRB in InterLock. For a 8×8 keyRB, we have $2log_2(m)-2 = 2(3)-2 = 4$ layers of SwBs. So, the timing paths must be the length of 4, and Fig. 4(a) shows a part of the timing path with a length of 4 that is selected to be embedded into the keyRB. Fig. 4(b) shows how this timing path is embedded into the keyRB. By using this approach, we embed m timing paths into a $m \times m$ keyRB allowing us to utilize the logic gates of each keyRB by 100%. Fig. 4(c) shows the top view of 8 different paths that are embedded into a keyRB while an arbitrary key has determined the connection between keyRB I/O.

4.1.3 *Twisted Logic in Interlock vs. Full-Lock*

In Full-Lock [17], it is claimed that by adding a layer of configurable inverters into each SwB, the logic could be twisted with the keyRB. For example, Fig. 2 shows that gates g_1 and g_6 are converted to its negated model ({*AND, NOT*} → {*NAND, BUFF*}, and the inversion is handled within keyRB. Hence, to handle the inversion inside each keyRB, a layer of inversion is added into each SwB. However, such usage of the sequence of key-programmable inversion does not elevate the security of Full-Lock, and the KeyRB of Full-Lock could be simplified. More precisely, to attack the Full-lock, one could remove all inverters from the KeyRB (from all layers), and just add one layer of the key-programmable inverters at the end to reduce the key size while still maintaining the same function.

[3]For a permutation-based $m \times m$ network, there are m different timing paths.

[4]The number of layers depends on the topology and being a blocking/non-blocking network. In this work, we use $2log_2(m) - 2$ layers of SwBs for $m \times m$ network [17] that builds a near non-blocking permutation network.

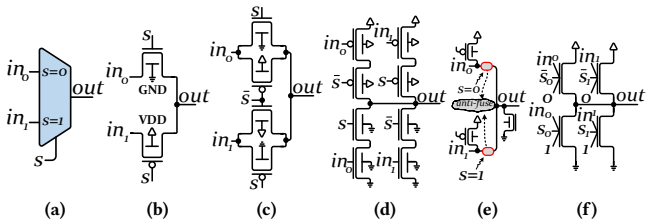


Figure 5: Different Multiplexer Implementation Possibilities: (a) 2:1 MUX Symbol, (b) 2:1 Pass-Transistor CMOS MUX, (c) 2:1 Transmission-Gate CMOS MUX, (d) 2:1 Static CMOS MUX, (e) 2:1 Anti-fused MUX, and (f) 2:1 TIGFET MUX.

This decouples the inversion from the routing block. Hence, in Full-Lock, the logic (inversion) and routing are not truly twisted. This allows us to simplify the KeyRB of Full-Lock before applying our *CP&SAT* attack to give maximum efficiency to the BVA. However, InterLock does not allow such simplification as f_1 and f_2 functions in each SwB are 2-input logic gates, and each input is (or could be considered as a) random and independent input.

4.2 Area/Delay Overhead Exploration

At first glance, embedding routing-based obfuscation incurs prohibited area/delay overhead. However, both Full-Lock and Cross-Lock engages some techniques to reduce the overhead to a reasonable ratio. Full-Lock shows how LUT insertion succeeding each keyRB allows them to use a smaller size of the keyRB to guarantee the resiliency at lower overhead. Unlike Full-lock that is implemented at gate-level and based on static CMOS technology, Cross-Lock engages anti-fuse-based elements called programmable via (PVIA) elements to minimize the overhead ratio of each keyRB. To implement InterLock in this paper, we examine both CMOS-based and PVIA-based implementation of keyRBs. Since MUXes are the only gate types that are used (overhead) for InterLock implementation, for CMOS-based implementation, amongst static logic, pass-transistor, or transmission gates, as demonstrated in Fig. 5(a)-5(d), we engage transmission-gates (Tgate) for MUXes based on tree-like structure [13, 22] that incur much lower overhead compared to static CMOS implementation. Also, as shown in Fig. 5(e), by using one-time-programmable elements (called PVIA elements in Cross-Lock [32]), we investigate the overhead of InterLock while implemented using anti-fuse-based (PVIA-based) 2:1 MUX. Similar to Tgate CMOS technology, we would use the tree-like structure to build keyRBs using PVIA elements.

Apart from these two technologies, in this paper, we assess the efficiency of another technology, called *Three-Independent-Gate Field Effect Transistors (TIGFET)*, for implementing MUXes in InterLock. In TIGFET technology, each transistor has *three* independent gates, and any two CMOS transistors could be modeled using only one TIGFET transistors, compacting the structure and achieving area as well as energy reduction, particularly for MUXes. Fig. 5(f) shows a 2:1 TIGFET multiplexer, and comparing with static CMOS each driving path consists of only one TIGFET transistors.

As shown in Fig. 6(a), Three terminal gates in TIGFET transistors called *Control Gate* (CG), *Polarity Gate at Drain* (PGD), and *Polarity Gate at Source* (PGS). Based on the value of these terminals, as illustrated in Fig. 6(b, c), one could build 2 series nFETs or 2-series pFETs. Since MUXes could be built using tristate inverters, in Fig. 6(d), we show how a tristate inverter could be built using TIGFET transistors. Compared to CMOS-based tristate inverter, the number of transistors is reduced by 50% in the TIGFET version. When m

tristate TIGFET inverters are cascaded, a $m:1$ MUX is built (e.g. 2:1 TIGFET MUX in Fig. 6(e)). It is worth mentioning that since the tristate inverter is used for each path of the multiplexer, the control signal (MUX selector) needs to be decoded. Hence, since all MUXes are controlled by the keys, and since we only use 2:1 MUXes, decoding selectors doubles the number of selectors (key inputs) in this technology. In our experimental results, we compare the implementation and the overhead of all three technologies.

5 EXPERIMENTAL RESULTS

To evaluate our proposed *CP&SAT* attack, we engage well-known ISCAS-89 and ITC-99 combinational circuits locked using Full-Lock [17][5][6]. We sweep the size of keyRBs to show the efficiency of the BVA algorithm on routing-based obfuscation. Further, for our proposed *InterLock*, as a countermeasure, we implement keyRBs from Fig. 3(b) on the same benchmark circuits to acquire locked circuits. We apply both the SAT (CycSAT-I) and our proposed *CP&SAT* attack on locked circuits by InterLock. All the experiments are implemented using Python/C++ and have been carried out on many Dell PowerEdge R620 equipped with Intel Xeon E5-2670 2.50GHz and 64GB of RAM. We evaluate the overhead of InterLock in three different technologies: (1) Transmission-Gate (Tgate) CMOS using Synopsys generic 32nm library, (2) PVIA-based MUXes that are manually added between the M2 and M3 layers as physical-only cells, and (3) Silicon NanoWire TIGFETs (TIG SiNWFETs 32nm) modeled using Verilog-A [12, 27].

5.1 The Efficiency of the BVA

To show how the BVA algorithm efficiently reduces the CNF size of the routing-based obfuscated circuits, Table 2 shows the rate of reduction that is more than a factor of two. The BVA adds/substitute the variables to decrease the number of clauses. As shown, the number of variables increases (by up to 2x); however, the number of clauses that play an important role in determining the complexity of the CNF is decreased by more than 2x. Hence, the BVA-based pre-processed CNFs are far easier for the SAT solver to be solved. Furthermore, as can be seen in Table 2, increasing the size of the keyRBs does not increase the SAT runtime after BVA pre-processing exponentially (most likely quadratically). Since we cannot infinitely increase the size of the keyRB (due to overhead and limitation of candidate selection), we report the results on keyRB with size up to 64×64.

To show the success of our proposed *CP&SAT* attack on routing-based obfuscation, in Table 3, we illustrate the runtime of the SAT (CycSAT-I) and the *CP&SAT* attack on circuits locked by Full-Lock.

[5]Since Cross-Lock is a weaker version of Full-Lock (It has no configurable inverters), we only report the attack results on circuits locked by Full-Lock.

[6]Since all ISCAS-89 and ITC-99 are sequential, we apply all techniques on combinational parts of these circuits, and we assume that all FFs are accesible to be read/written for both SAT and *CP&SAT* attack.

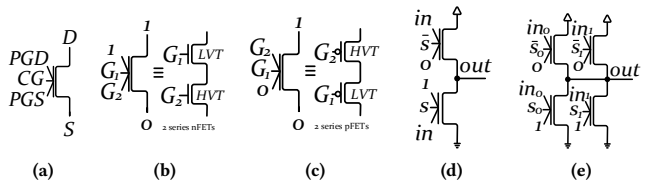


Figure 6: 2:1 TIGFET MUX Implementation: (a) A TIGFET Transistor, (b) 2 Series nFETs with One TIGFET Transistor, (c) 2 Series pFETs with One TIGFET Transistor, (d) A Tristate Inverter built by TIGFET Transistors, (e) 2:1 TIGFET MUX by Cascading TIGFET Tristate Inverters.

Table 2: The Effectiveness of the BVA Pre-Processing Step on Different-Size keyRBs in Routing-based Obfuscation.

Instance	Original			BVA Pre-processed		
	#Variables	#Clauses	Solve	#Variables	#Clauses	pre+Solve
keyRB-4	271	418	**0.02**	428	202	**0.01+0.22**
keyRB-8	875	1606	**0.45**	1278	718	**0.01+0.36**
keyRB-12	1544	3084	**2.48**	2188	1288	**0.01+0.54**
keyRB-16	2419	4750	**5.42**	3982	2184	**0.01+0.82**
keyRB-24	3372	7502	**54.82**	4618	3452	**0.02+1.64**
keyRB-32	6178	12510	**194.8**	8892	7258	**0.02+2.22**
keyRB-48	9891	18614	**Timeout**	12672	9918	**0.04+3.92**
keyRB-64	15043	31182	**Timeout**	23818	14772	**0.04+12.22**

Timeout = 10^5 *Seconds* ≈ *1 day*

As shown, in all cases, after inserting four keyRB-16 (16×16), the traditional SAT attack fails to break the locked circuits. However, when we apply the *CP&SAT* attack, all circuits locked by four keyRB-16 are broken in less than 10 minutes. When we assume that the configurable inverters of SwBs in Full-Lock are in place, the BVA algorithm within the *CP&SAT* attack does not provide a significant advantage. As shown in Table 3, we observed that this new attack also fails to break circuits locked with four keyRB-16 while inverters are intact. However, as described previously, we detach the inverters in Full-Lock by fixating the key values of all layers of inverters (disable the inversion) except the last layer. When the inverters are detached, the BVA could efficiently reduce the size of the locked circuit allowing us to de-obfuscate within few minutes.

Table 3: The Runtime of the SAT Attack and the *Cnonical Prune-and-SAT* Attack on Circuits Locked by Full-Lock [17] with Different Sizes of KeyRBs.

Circuit	#Gates	#I/O	Traditional SAT Attack (CycSAT-I)			proposed *canonical prune&SAT* (with inverters)			proposed *canonical prune&SAT* (detached inverters)		
			keyRB-16 (16×16)			keyRB-16			keyRB-16		
			2	3	4	2	3	4	2	3	4
b15	~8.5K	485/519	1507.5	TO	TO	486.4	2581.5	TO	44.8	75.9	319.8
b14	~9.5K	277/299	788.3	TO	TO	329.7	1688.8	TO	34.8	88.9	416.6
s35932	~16K	1763/2048	856.6	TO	TO	643.8	5238.1	TO	76.4	147.9	407.4
s38417	~18K	1464/1731	1187.4	TO	TO	482.5	2037.9	TO	58.2	100.7	366.3
b20	~19.5K	522/512	1096.8	TO	TO	537.8	3507.9	TO	70.8	129.4	411.2
b21	~20K	522/512	1832.4	13283	TO	984.8	8207.3	TO	134.4	207.8	550.1
b17	~30K	1452/1512	508.2	8401.7	TO	306.8	6095.4	TO	81.7	137.4	463.8
b22	~30K	767/757	924.6	6491.5	TO	508.2	5538.4	TO	68.5	99.1	390.5
b18	~110K	3357/3343	1283.7	9208.1	TO	581.9	6327.8	TO	91.6	162.7	472.2

Timeout (TO) = 10^5 *Seconds* ≈ *1 day*

5.2 Disabling the BVA using InterLock

To show how InterLock could be used as a countermeasure against the *Canonical prune and SAT* attack, we insert different numbers of the keyRBs from Fig. 3(b) into the benchmark circuits with different sizes. For the insertion of the keyRBs two strategies have been considered/applied in InterLock: (1) To minimize the performance degradation (delay overhead), the timing paths that are selected as the candidates to be embedded into keyRBs must be one of the highest positive slack timing paths, and (2) as shown in Table 1, since having more *XNOR*s (*XOR*s) increase the resilience of the locked circuits considerably, amongst the candidates, we select those paths that have more *XNOR*s (*XOR*s).

Table 4 shows the runtime of both the SAT and the *canonical prune and SAT* attack on circuits locked by InterLock. In both cases, since the locked circuit is possibly cyclic, for the SAT solving part, CycSAT-I has been used. As shown, for almost all cases, after inserting only two keyRB-16, both attacks fail to break the locked

Table 4: The Runtime of the SAT Attack as well as the *Cnonical Prune-and-SAT* Attack on Circuits Locked by InterLock with Different Sizes of KeyRBs.

Circuit	Traditional SAT Attack (CycSAT-I)					proposed *canonical prune&SAT*				
	keyRB-8			keyRB-16		keyRB-8			keyRB-16	
	1	2	3	1	2	1	2	3	1	2
b15	98.5	3807.8	TO	74127.8	TO	85.3	3207.8	TO	69328.5	TO
b14	120.8	4229.1	TO	67203.2	TO	105.1	4028.5	TO	64328.6	TO
s35932	291.7	7126.4	TO	59372.1	TO	260.7	6992.1	TO	55221.4	TO
s38584	267.9	7624.4	TO	71375.5	TO	233.8	7168.5	TO	63298.8	TO
b20	284.4	11275.8	TO	58348.6	TO	186.7	8673.8	TO	55373.3	TO
b21	738.4	TO	TO	TO	TO	672.5	TO	TO	TO	TO
b17	320.4	6221.3	TO	77023.3	TO	271.9	5882.2	TO	74623.7	TO
b22	376.4	5209.9	TO	51042.4	TO	126.7	3862.7	TO	37621.2	TO
b18	701.9	32841.5	TO	TO	TO	630.3	30067.7	TO	TO	TO

Table 5: The Average Number of Iterations for De-Obfuscating Different Sizes of KeyRBs. (Numbers in parentheses show the Current Iteration at Timeout).

Insance / Model	keyRB-4	keyRB-8	keyRB-16	keyRB-32	keyRB-64
Full-Lock [17]	3-5	4-6	8-10	10-12	(5) Timeout
InterLock	8-12	16-22	30-33	(8) Timeout	(9) Timeout

circuit. Unlike previous routing-based obfuscation techniques, BVA does not provide any advantage as a pre-processing step showing that truly twisting logic into the keyRBs guarantees the resistance against this new attack. Furthermore, compared to Full-Lock and Cross-Lock, twisting logic into keyRB allows us to engage smaller sizes of keyRB to guarantee the resiliency (keyRB-32/16 → keyRB-16/8). Shrinking the size of the keyRB with guaranteed security in InterLock allows the designer to considerably reduce area and delay overhead.

To illustrate that InterLock elevate the complexity of locked circuit, in Table 5, we compare the average number of iterations (N in Eq. 1) in Full-Lock with that of InterLock. Since we still get the benefit of routing-based obfuscation, the number (M in Eq. 2) and the computational complexity (T^{Avg}_{DPLL} in Eq. 2) of recursive calls in DPLL algorithm is still extremely high in InterLock. However, as illustrated in Table 5, the average number of required iterations is increased by ~3x-4x. This increase shows that $MN \times T^{Avg}_{DPLL}$ (from Eq. 2) is extremely higher in InterLock deepening the logic locking problem significantly.

5.3 Elevated Security at Lower Overhead

To perform a proof-of-concept physical design flow, we implement the keyRB in InterLock using three different 32nm technology: (1) Transmission-Gate (Tgate) CMOS, (2) PVIA-based MUXes, and (3) TIGFET SiNWFETs using Verilog-A. Table 6 shows the area/power/delay overhead of locked circuits via three keyRB-8, which is resilient against the SAT and the *cnonical prune-and-SAT* attack. Compared to Full-Lock which is a gate-level implementation of MUXes using static CMOS, in InterLock, Tgate-based implementation at transistor level would considerably reduce the area/delay overhead. In many cases it was expected to observe that PVIA-based MUXes could achieve the most optimum results; However, since there is no automatic flow in existing EDA tools for optimization of a large number of PVIA-based elements, the insertion has to be done manually. To do it manually, we inserted the PVIAs in a grid and push the standard cells away from this PVIA grid to successfully perform placement, and due to fine-granularity of MUXes in the circuit (small units and a large amount of usage), and since the number of PVIAs that must be used is a lot, in many cases we faced DRC violations leading us to use much lower utilization rate for them.

Table 6: The Overhead Comparison between Three Different Technology used for InterLock Implementation: Tgate, Anti-fuse, and TIGFET.

Circuit	Original			3×keyRB-8								
				Tgate CMOS			PVIA Anti-Fuse			TIGFET		
	a (μm^2)	p (μW)	d (ns)	a %	p %	d %	a %	p %	d %	a %	p %	d %
b15	5292.7	327.6	1.23	34.5%	27.8%	12.9%	27.6%	21.9%	7.1%	24.5%	20.1%	6.4%
b14	5707.9	423.9	1.55	22.6%	17.5%	14.6%	19.5%	16.2%	8.8%	18.6%	15.4	6.8%
s35932	9283.1	729.8	1.68	30.7%	22.8%	10.9%	27.3%	20.7%	6.5%	24.8%	18.7%	5.1%
s38584	11003.2	806.6	1.77	24.1%	19.1%	19.7%	22.4%	17.3%	10.7%	20.7%	16.7%	8.3%
b20	11752.5	755.6	2.34	19.8%	15.5%	12.8%	17.6%	13.9%	7.1%	15.9%	13.4%	5.6%
b21	13007.1	922.7	2.21	16.2%	12.7%	10.7%	14.3%	11.2%	5.7%	12.9%	10.4%	4.5%
b17	15573.3	1245.1	3.58	8.9%	7.4%	7.6%	7.8%	6.4%	4.2%	7.2%	5.9%	3.4%
b22	16582.7	1319.5	3.18	6.4%	4.9%	8.5%	5.9%	4.1%	4.3%	4.8%	3.9%	3.5%
b18	57626.9	4834.1	3.81	3.7%	2.1%	3.2%	3.5%	1.8%	1.9%	2.7%	1.4%	1.6%

Based on our evaluation of these three different technologies, as shown in Table 6, TIGFET-based keyRB could bring more efficiency compared to Tgate CMOS and PVIA-based implementation. As shown, on average, TIGFET could reduce the area/delay overhead by up to 20%/56% compared to Tgate-based CMOS keyRB. However, for two important reasons, in all three technologies, on average, the overhead is less than 10%, which is completely acceptable: (1) The required number/size of keyRBs is less/smaller in InterLock, and (2) The actual timing paths selected for embedding are paths with highest positive slack time.

6 CONCLUSION

In this paper, we proposed a new attack, *canonical prune-and-SAT* (*CP&SAT*) attack, for breaking the state-of-the-art routing-based obfuscation solutions (e.g. Full-Lock [17] and Cross-Lock [32]). In this attack, we exploited a *bounded variable addition* (BVA) pre-processing step (before the SAT attack) to reduce the size and complexity of the CNF representation of the key-programmable routing blocks (keyRBs) used for routing-based obfuscation. We demonstrated that by adding the BVA pre-processing step, our proposed attack reduces the size of the targeted CNF formula by a factor of two, enabling us to easily break both Full-Lock and Cross-Lock in a reasonable time. Further, as a countermeasure, we proposed our unified routing and logic obfuscation techniques, coined as *InterLock*. In Interlock, in addition to hiding the wiring/interconnection, a part of the logic (gates) in the selected timing paths are also implemented in the keyRB. We illustrated that when the logic gates are twisted with keyRBs, the BVA could not provide any advantage as a pre-processing step. We demonstrated that, by using InterLock, the resilience against existing attacks as well as our new proposed *CP&SAT* attack would be guaranteed while the delay/area overhead remains acceptable. We further evaluated and depicted the result of implementing the Interlock using three different technologies: Transmission-gate CMOS, anti-fuse elements, and three-independent-gate field-effect transistors (TIGFET).

ACKNOWLEDGEMENT

This research is supported by the Defense Advanced Research Projects Agency (DARPA-AFRL, #FA8650-18-1-7819), National Science Foundation (NSF, #1718434), and partly by Silicon Research Co. (SRC TaskID 2772.001).

REFERENCES

[1] A. Biere. 2004. Resolve and Expand. In *Int'l Conference on Theory and Applications of Satisfiability Testing*. 59–70.

[2] A. Biere *et al.* 2014. Detecting cardinality constraints in CNF. In *International Conference on Theory and Applications of Satisfiability Testing*. 285–301.

[3] A. Chakraborty *et al.* 2018. TimingSAT: timing profile embedded SAT attack. In *ICCAD*. 1–6.

[4] A. Rezaei *et al.* 2018. Cyclic Locking and Memristor-based Obfuscation against CycSAT and Foundry Attacks. In *DATE*. 85–90.

[5] A. Rezaei *et al.* 2019. CycSAT-unresolvable cyclic logic encryption using unreachable states. In *ASP-DAC*. 358–363.

[6] A. Yeh. 2012. Trends in the Global IC Design Service Market. *DIGITIMES research* (2012).

[7] Actel Corporation. 2002. Design Security in Nonvolatile Flash and Antifuse FPGAs. , 16 pages. http://www.actel.com/documents/DesignSecurity_WP.pdf

[8] D. Chai *et al.* 2005. A Fast Pseudo-Boolean Constraint Solver. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 3 (2005), 305–317.

[9] D. Le Berre *et al.* 2010. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation* 7, 2-3 (2010), 59–64.

[10] D. Mitchell *et al.* 1992. Hard and Easy Distributions of SAT Problems. In *AAAI*, Vol. 92. 459–465.

[11] D. Sirone, and P. Subramanayan. 2020. Functional analysis attacks on logic locking. *IEEE TIFS* (2020).

[12] E. Giacomin *et al.* 2017. Low-power Multiplexer Designs using Three-Independent-Gate Field Effect Transistors. In *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. 33–38.

[13] E. Lee *et al.* 2008. Interconnect Driver Design for Long Wires in Field-Programmable Gate Arrays. *Journal of Signal Processing Systems* 51, 1 (2008), 57–76.

[14] F. Aloul *et al.* 2002. Solving Difficult SAT Instances in the Presence of Symmetry. In *Design Automation Conference (DAC)*. 731–736.

[15] G. -J. Nam *et al.* 2004. A Comparative Study of Two Boolean Formulations of FPGA detailed Routing Constraints. *IEEE Trans. Comput.* 53, 6 (2004), 688–696.

[16] G. L. Zhang *et al.* 2018. TimingCamouflage: Improving Circuit Security against Counterfeiting by Unconventional Timing. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 91–96.

[17] H. M. Kamali *et al.* 2019. Full-Lock: Hard Distributions of SAT Instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks. In *DAC*.

[18] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan. 2020. On Designing Secure and Robust Scan Chain for Protecting Obfuscated Logic. In *Great Lakes Symposium on VLSI (GLSVLSI)*. 1–6.

[19] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan. 2020. SCRAMBLE: The State, Connectivity and Routing Augmentation Model for Building Logic Encryption. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 153–159.

[20] H. M. Kamali, K. Z. Azar, K. Gaj, H. Homayoun, and A. Sasan. 2018. LUT-lock: A Novel LUT-based Logic Obfuscation for FPGA-bitstream and ASIC-hardware Protection. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 405–410.

[21] H. Zhou *et al.* 2017. CycSAT: SAT-based Attack on Cyclic Logic Encryptions. In *ICCAD*. 49–56.

[22] J. Luu *et al.* 2014. VTR 7.0: Next Generation Architecture and CAD System for FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 7, 2 (2014), 1–30.

[23] J. Marques-Silva *et al.* 2007. Towards robust CNF encodings of cardinality constraints. In *International Conference on Principles and Practice of Constraint Programming*. 483–497.

[24] J. Rajendran *et al.* 2012. Security Analysis of Logic Obfuscation. In *Proceedings of Design Automation Conference (DAC)*. 83–89.

[25] J. Rajendran *et al.* 2015. Fault Analysis-based Logic Encryption. *IEEE TC* 64, 2 (2015), 410–424.

[26] J. Roy *et al.* 2010. Ending Piracy of Integrated Circuits. *Computer* 43, 10 (2010), 30–38.

[27] J. Zhang *et al.* 2014. Configurable Circuits Featuring Dual-Threshold-Voltage Design with Three-Independent-Gate Silicon Nanowire FETs. *IEEE Transactions on Circuits and Systems I: Regular Papers* 61, 10 (2014), 2851–2861.

[28] K. Shamsi *et al.* 2017. AppSAT: Approximately Deobfuscating Integrated Circuits. In *HOST*. 95–100.

[29] K. Shamsi *et al.* 2017. Cyclic Obfuscation for Creating SAT-Unresolvable Circuits. In *GLSVLSI*. 173–178.

[30] K. Shamsi *et al.* 2019. IcySAT: Improved SAT-based Attacks on Cyclic Locked Circuits. In *ICCAD*. 1–7.

[31] K. Shamsi *et al.* 2019. KC2: Key-condition Crunching for Fast Sequential Circuit Deobfuscation. In *DATE*. 534–539.

[32] K. Shamsi *et al.* 2018. Cross-lock: Dense layout-level interconnect locking using cross-bar architectures. In *GLSVLSI*. 147–152.

[33] K. Z. Azar *et al.* 2019. COMA: Communication and Obfuscation Management Architecture. In *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. 181–195.

[34] K. Z. Azar *et al.* 2019. SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks. *CHES* (2019), 97–122.

[35] K. Z. Azar, H. M. Kamali, H. Homayoun, A. Sasan. 2020. NNgSAT: Neural Network guided SAT Attackon Logic Locked Complex Structures. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9.

[36] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan. 2019. Threats on logic locking: A decade later. In *Great Lakes Symposium on VLSI (GLSVLSI)*. 471–476.

[37] L. Alrahis *et al.* 2019. ScanSAT: Unlocking Obfuscated Scan Chains. In *ASP-DAC*. 352–357.

[38] M. Alam *et al.* 2019. TOIC: Timing Obfuscated Integrated Circuits. In *Great Lakes Symposium on VLSI*. 105–110.

[39] M. Davis *et al.* 1960. A Computing Procedure for Quantification Theory. *Journal of the ACM (JACM)* 7, 3 (1960), 201–215.

[40] M. El Massad *et al.* 2015. IC Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes. In *Network and Distributed System Security Symposium (NDSS)*. 1–14.

[41] M. El Massad *et al.* 2017. Reverse Engineering Camouflaged Sequential Circuits without Scan Access. In *ICCAD*. 33–40.

[42] M. Gebser *et al.* 2009. The conflict-driven answer set solver clasp: Progress report. In *International conference on logic programming and nonmonotonic reasoning*. 509–514.

[43] M. Koshimura *et al.* 2012. QMaxSAT: A partial Max-SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation* 8, 1-2 (2012), 95–100.

[44] M. Rostami, F. Koushanfar, and R. Karri. 2014. A Primer on Hardware Security: Models, Methods, and Metrics. *Proc. IEEE* 102, 8 (2014), 1283–1295.

[45] M. Velev *et al.* 2008. Comparison of Boolean Satisfiability Encodings on FPGA detailed Routing Problems. In *Design, Automation and Test in Europe*. 1268–1273.

[46] M. Velev *et al.* 2009. Efficient SAT Techniques for Absolute Encoding of Permutation Problems: Application to Hamiltonian Cycles. In *Eighth Symposium on Abstraction, Reformulation, and Approximation*.

[47] M. Yasin *et al.* 2016. SARLock: SAT Attack Resistant Logic Locking. In *HOST*. 236–241.

[48] M. Yasin *et al.* 2017. Provably-Secure Logic Locking: From Theory to Practice. In *ACM CCS*. 1601–1618.

[49] M. Yasin *et al.* 2017. Removal Attacks on Logic Locking and Camouflaging Techniques. *IEEE TETC* (2017).

[50] M. Yasin *et al.* 2017. Security Analysis of Anti-SAT. In *ASP-DAC*. 342–347.

[51] N. Eén *et al.* 2003. An extensible SAT-solver. In *International conference on theory and applications of satisfiability testing*. 502–518.

[52] N. Eén *et al.* 2005. Effective preprocessing in SAT through variable and clause elimination. In *Int'l conference on theory and applications of satisfiability testing*. 61–75.

[53] N. Limaye *et al.* 2019. Is Robust Design-for-Security Robust Enough? Attack on Locked Circuits with Restricted Scan. In *ICCAD*. 1–8.

[54] N. Manthey *et al.* 2012. Automated Reencoding of Boolean Formulas. In *Haifa Verification Conference*. 102–117.

[55] O. Bailleux *et al.* 2003. Efficient CNF encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*. 108–122.

[56] P. Subramanyan *et al.* 2015. Evaluating the Security of Logic Encryption Algorithms. In *International Symposium on Hardware Oriented Security and Trust (HOST)*. 137–143.

[57] P. Tuyls *et al.* 2006. Read-Proof Hardware from Protective Coatings. In *Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. 369–383.

[58] R. Asín *et al.* 2011. Cardinality networks: a theoretical and empirical study. *Constraints* 16, 2 (2011), 195–221.

[59] R. Karmakar *et al.* 2018. Encrypt flip-flop: A novel logic encryption technique for sequential circuits. *arXiv 1801.04961* (2018).

[60] S. Patnaik *et al.* 2020. Obfuscating the Interconnects: Low-cost and Resilient Full-chip Layout Camouflaging. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2020).

[61] S. Roshanisefat *et al.* 2018. SRCLock: SAT-resistant cyclic logic locking for protecting the hardware. In *Great Lakes Symposium on VLSI (GLSVLSI)*. 153–158.

[62] S. Roshanisefat *et al.* 2020. SAT-Hard Cyclic Logic Obfuscation for Protecting the IP in the Manufacturing Supply Chain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2020), 1–14.

[63] S. Roshanisefat, H. M. Kamali, K. Z. Azar, S. M. P. Dinakarrao, N. Karimi, H. Homayoun, and A. Sasan. 2020. DFSSD: Deep Faults and Shallow State Duality, A Provably Strong Obfuscation Solution for Circuits with Restricted Access to Scan Chain. In *VLSI Test Symposium (VTS)*. 1–6.

[64] C. Sinz. 2005. Towards an optimal CNF encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*. 827–831.

[65] X. Wang *et al.* 2017. Secure Scan and Test using Obfuscation throughout Supply Chain. *IEEE TCAD* 37, 9 (2017), 1867–1880.

[66] X. Xu *et al.* 2017. Novel Bypass Attack and BDD-based Trade-off against all Known Logic Locking Attacks. In *CHES*. 189–210.

[67] Y. Shen and H. Zhou. 2017. Double Dip: Re-evaluating Security of Logic Encryption Algorithms. In *GLSVLSI*. 179–184.

[68] Y. Shen *et al.* 2019. BeSAT: behavioral SAT-based attack on cyclic logic encryption. In *ASP-DAC*. 657–662.

[69] Y. Xie, and A. Srivastava. 2016. Mitigating SAT Attack on Logic Locking. In *CHES*. 127–146.

[70] Y. Xie and A. Srivastava. 2017. Delay Locking: Security Enhancement of Logic Locking against IC Counterfeiting. In *DAC*. 1–9.