INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

Hardware Logic Locking Obfuscation for Cyber Hygiene Based on Hamming Distance Obfuscator

Akhigbe-mudu Thursday Ehis
Department of Computer Science, Faculty of Applied Sciences and Engineering, African Institute of Science administration and Commercial Studies Lome-Republic of Togo
akhigbe-mudut@iaec-university.tg

----------***----------

Annotation: The circuit netlist, logic locking mode, is an obfuscation method used to protect outsourced chip designs. Logic locking has been demonstrated to be broken via Boolean Satisfiability-based attacks, which has spurred researchers to create more robust defenses. The development of SAT attack marked a turning point in research on logic locking. Software systems frequently update their huge executable code, so obfuscating the program for each small update results in a significant loss of efficiency. In order to provide security guarantees against synthesis-driven assaults, this study provides a transformation strategy. Its Random Technique introduces a novel fault injection attack to undermine any locking mechanism that depends on a saved private key. This made it possible to compare the obfuscated circuit to its source, to determine whether a sufficient structural change has been made to support its functionality. The RT created an attack strategy to discover the value of the private key, K*. It examines the SFLL's security and offered a solution to determine the hamming distance, "h," using one or more SAT queries. It proposes an effective bit-flipping attack using the irregularity between the protected input patterns and the private key. By flipping bits, the attack cracks the private key, K*, with (n-1) queries. The outcome demonstrates that, given a protected input pattern, the right key may be quickly discovered by bit-flipping.

Keywords: Algorithms, Flipping, Logic Locking, Random Technique, Satisfiability.

1.0 Introduction

A special type of hardware obfuscation known as "logic locking" involves adding extra logic components or key gates to a netlist [48]. Key gates boost the security and privacy of the hardware while only adding a modest amount of extra cost to the circuit design. The security against particular sorts of attacks is determined by the insertion method and algorithm of key gates. If a circuit is locked, a special key must be used to unlock it; otherwise, the circuit will behave incorrectly when it outputs data. Techniques for logic locking encryption can be developed to defend against certain attacks such side channel attacks, key sensitization attacks, and SAT attacks.

Additionally, researchers have merged concepts from several methodologies to produce stronger and more secure netlists [20]. In order to make it more difficult for an attacker to reverse-engineer a circuit, a technique called hardware obfuscation alters the description or structure of a circuit. Some obfuscation methods alter the circuit's gate level layout, while others add gates to shield the circuit's logic [3]. By adding extra gates and logic elements to a circuit, logic locking is a technique that locks the circuit and causes an inaccurate output unless the right key is supplied to the circuit. Until the additional gates are unlocked with the right key, the Integrated Circuit (IC) will be regarded as locked or functionally wrong. The correct key value will cause the gate to behave as a buffer and have no impact on the remaining logic when XOR and XNOR components are used as key gates. The key gate will produce the incorrect value if the incorrect key value is supplied, rendering the circuit inoperable. An overview of the logic locking technique is shown in Figure 1. The netlist of the original circuit is enhanced by adding more key gates and a key-value to unlock the circuit. The most common approach to addressing the dangers from untrusted manufacture has evolved, is logic locking [10], [34], [9] and [22].

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071

Volume: 4 Issue: 10 | October 2022

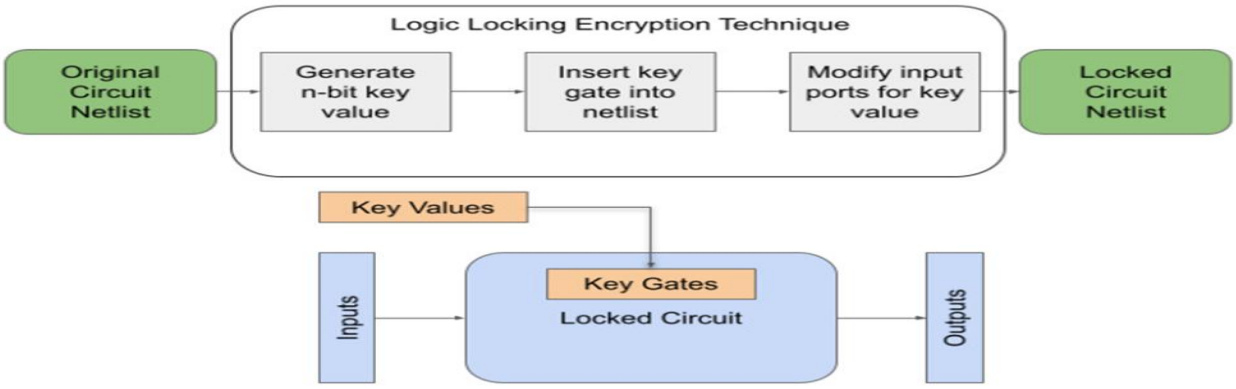

Figure 1: An Over view of Logic Locking technique

In logic locking, a circuit's netlist is secured with a secret key, preventing the circuit from operating as intended. Figure 2 depicts an abstract representation of logic locking, in which the key is kept in memory and used to open the locked circuit's functionality. The key must be kept a secret, and caution must be exercised during the design phase so that information is not immediately leaked during operations to the principal output. Between logic gates, a key gate is inserted, with one input coupled to the value of the key bit. These key gates increase the device's security while introducing a minor overhead to the circuit as a whole [5] [22].

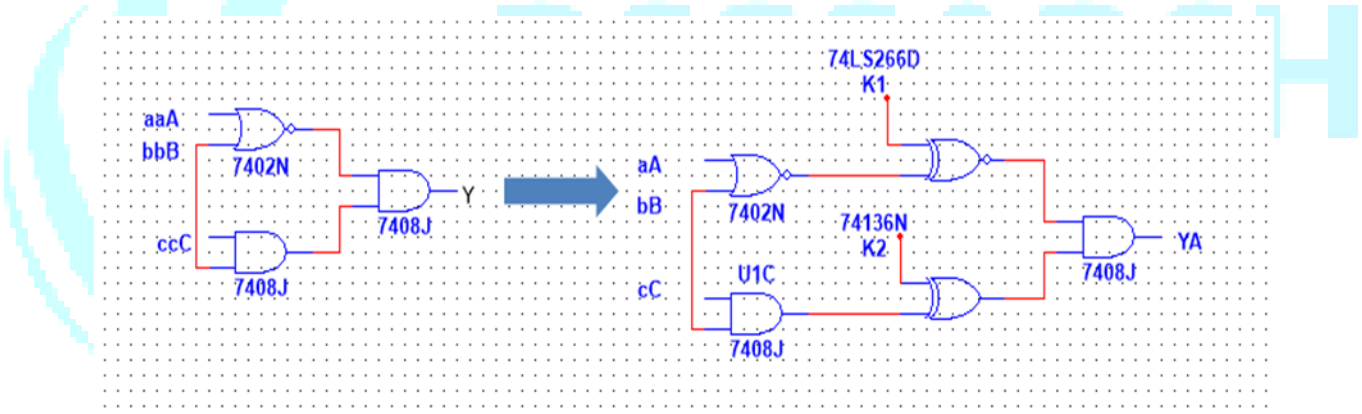

Figure 2: Simplified Example of Logic Locking Method

1.1.0 Fundamental of Logic Locking

Figure 2 displays an easy illustration of logic locking. Two-input OR and AND gates make up the original logic gate. To further combine the initial output f with a locking enable signal k, an exclusive-OR gate is added. Next, there are two logic XOR gates in the locked netlist, in that order. Equation (1) gives the locked Boolean logic function**.**

$$f_{lock} = f.\bar{k} + \bar{f}.k \qquad (1)$$

Where $f = ab$, f is the original function, k is the locking enable and f_{lock} is the locked function. An application-specific integrated circuit (IC) or processor with dedicated security hardware is used in hardware-based security to perform encryption functions and thwart assaults. At the IC hardware level, where encryption algorithm performance is improved, security operations like encryption or decryption and authentication take place (JeanPaul et al., 2020). Additionally, sensitive data is shielded inside the electrical confines of the encryption hardware, including keys and crucial final application parameters. The protection of consumer data and the

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

creators' intellectual property were among the various countermeasures for physical attacks that were described in earlier literature in order to secure an IoT system.

Integrated circuits (ICs) can be secured using the hardware security method of logic encryption by adding extra gates. By ensuring that predetermined outputs are only produced when the right key inputs are given, the inserted gates stop IC forgery, IP theft, and IC overproduction. Two main criteria are typically used to assess the robustness of logic encryption: (1) interdependency between keys, and (2) output corruption against assaults such path sensitization attack, SAT based attack, hill-climbing attack, etc. However, the vast majority of logic encryption techniques in use prioritize one requirement over another. This paper proposes a fully correlated key interdependency block enhanced logic encryption approach. The technique uses an uncommon node analysis method to pinpoint the positions of key-gates and strengthens the interdependency of keys.

1.1.1 Classification of logic locking

Traditional logic locking and SAT attack resistant logic locking are the two main forms of logic locking.

1.1.2 Traditional logic locking.

The methods in this group were concentrated on creating effective algorithms for choosing the key gate locations. Strong logic locking [36] and random logic locking [35] are three examples of classical logic locking strategies. Sequential logic locking methods include [52], [5], [6], [8], [9], [10].

1.1.3. Resilient logic locking against SAT attacks.

The SAT attack, which was able to defeat all conventional logic locking methods, changed the course of logic locking research completely [25]. Recent research initiatives like SARLock [39], Anti-SAT [3], TTLock [9], and SFLL [12], focus on thwarting the SAT attack and ensure 50% Hamming distance with Logic locking metric

1.1.4. Logic locking metric

A protection method based on the addition of extra logic gates has its fundamental component as the ability to modify outputs [1]. Therefore, two qualities can be utilized to assess the efficacy of these strategies. The first one is: how many inputs gate keys can span each extra logic gate? This has to do with how many gates need to be installed to enable complete functional locking. It is plainly more efficient to lock several outputs with a single gate rather than using multiple gates. Following is a definition of the locking ratio:

$$Locking.Ratio = \frac{\#outputs}{\#locking.gates} \qquad (2)$$

A second indicator is how far the inserted gate is from the outputs since locking gates should be inserted as deeply into the netlist as possible [49]. As a result, the calculation also includes the number of logic levels between the locking gate and the outputs. The average number of logic levels on the shortest path between the inserted gates and each output that can be reached is used to calculate the average distance between the inserted gates and the outputs [31].

1.1.5. Obfuscation

Digital Signal Processing (DSP) plays a critical role in numerous applications such as video compression, portable systems/computers, multimedia, wired and wireless communications, speech processing and biomedical signal processing [47]. However, the security aspect for DSP applications has only attracted little attention in the literature. While PUFs can be used as authentication-based methods to improve the security of DSP circuits, obfuscation-based approaches are also obliged to protect the intellectual property. Design obfuscation is a technique that transforms an application or a design into one that is functionally equivalent to the original but is

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

significantly more difficult to reverse engineer. In this study, a novel design methodology for obfuscated DSP circuits is designed by hiding functionality via high-level transformations [26]. The DSP circuits are obfuscated by introducing a finite-state machine (FSM) whose state is controlled by a key. The FSM enables a reconfigurator that configures the functionality mode of the DSP circuit [38]. High-level changes lead to numerous proportionate circuits and all these create equivocalness within the structural level. Also, high-level changes permit plan of circuits utilizing same information but diverse control-flows. Diverse variety modes can be embedded into the DSP circuits for obscurity. Some modes produce functionally incorrect outputs, but the outputs are meaningful from a signal processing point of view, so they can represent the correct output in many situations. Other modes lead to meaningless output. The initialization key and configuration data must be known for the circuit to work correctly [40]. As a result, the proposed design methodology produces DSP circuits that are both structurally and functionally obfuscated. High-level transforms have been leveraged for their speed-performance tradeoffs in the domain, but our work is the first to leverage the security perspective of high-level transforms.

1.1.6 What is Hamming Distance?

The Hamming distance sum up the comparing components that vary between two vectors. In practical terms, the more noteworthy the Hamming distance, the more the two vectors contrast. Contrarily, the smaller the Hamming distance, the more comparable the two vectors are [19].

Mathematically, the Hamming Distance is represented by the formula below:

$$d(x, y) = \frac{1}{n} \sum_{n=1}^{n=n} |x_i - y_i| \quad (3)$$

1.1.6.1 Complete Code:

```
public class HummingDistanceString {
public static void get Distance(String x, String y){
int humming_distance =0;
if(x.length()!=y.length()){
System.out.println("Both string sizes are different");
return;
}
for (int i = 0; i <x.length() ; i++) {
if(x.charAt(i)!=y.charAt(i))
humming_distance++;
}
System.out.println("x="+x+", y="+y+" Hamming distance: " + humming_distance);
}
public static void main(String[] args) {
String x = "AABBCCDD";
String y = "AAAACCCC";
```

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

getDistance(x, y);

x = "dogandcat";

y = "catanddog";

getDistance(x, y);

}

}

Output:

x=AABBCCDD, y=AAAACCCC Hamming distance: 4

x=dogandcat, y=catanddog Hamming distance: 6

To run the analysis, a key bit matrix of (say) size 2|B| × |B| is generated while each row represents a unique hypothesis of |B| key bits. As previously mentioned, it's expected that challenges which is in the same hypothesis decision, that is, same key bit value (say, B), shows a small hamming distance. So for each hypothesis decision, the pairwise hamming distances generate the same key bit ($B_i = B_j$), summed up and form a fitness value f. For example, for |B| = 5 and key hypothesis $B_1 \ldots 5 = \{1, 1, 0, 1, 0\}$ the fitness value:

$$f = f_{ones} + f_{zeroes}$$

$$= HD(c_1, c_2) + HD(c_1, c_4) + HD(c_2, c_4) + HD(c_3, c_5) \qquad (4)$$

The fitness value of such a hypothesis is directly related to the probability that this hypothesis is the correct key. The lower the fitness value f of a hypothesis, the more likely the hypothesis is the correct one. To quantify the results of the hamming distance characterization, a well-known definition of Shannon **Entropy** would be used [5]**.**

1.1.6.2. Shannon Entropy

Entropy, specifically relates to Shannon Entropy, measures the amount of information present in a source of data. For a combinational circuit (or equivalently, a Boolean function), entropy relates to the number of distinct outputs that can be produced by the function [33]. For a single output, Boolean function with a probability of logic-1 is denoted by P_i, the entropy is given by Eq. 5:

$$H = -\sum_{i=1}^{2^{|B|}} p_i \log_2 p_i \qquad (5)$$

Let p_i be the probability that the correct key bit hypothesis is placed at position i in the ranked list generated by the hamming distance characterization. The Probability Density Function (PDF) for the entropy estimation is empirically generated from the runs and is therefore just an approximation. Increasing the runs had no influence on the PDF, so it is assumed that a sufficient accurate PDF is generated..

1.1.7.0. Statement of the Problem

In the context of cyber security, obfuscation is a method of manipulating a computer program with the intention to obscure its inner workings [28]. Various obfuscation techniques have found their use as a means to protect intellectual property and prevent code tampering, as well as malicious purposes, such as creating malware that can circumvent detection mechanisms [2]. Despite their numerous advantages, hardware obfuscation is prone to various cyber threats, attacks and challenges.

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

Recently, a logic locking technique referred to as stripped functionality logic locking (SFLL) has been projected and shown to face familiar attacks in an exceedingly incontrovertibly secure manner [25]. SFLL strips some functionality from the initial design by corrupting its output to a variety of "protected" input patterns. This current SFLL implementation depends on the present security agnostic tools like Synopsys DC Compiler [37], for synthesis. Netlists synthesis using these tools may leave behind traces for an attacker to exploit. This work plans to modify the tools and offer security guarantees against synthesis-driven attacks [32], [39].

Secondly, the SFLL relies on the designer to decide which parts of the design is safety critical. This work automates this process by developing metrics that quantify the importance of various circuit components and identify protected patterns that require minimal implementation effort and saves power, or latency.

As discussed previously, the introduction of the SAT attack could be considered as the turning point in logic locking. However, software systems typically contain large amounts of executable codes that are updated severally, sometimes very frequently. So re-obfuscating the program after every little update would lead to a considerable loss of efficiency. Therefore, this study presents a transformation design to strengthen security from weaker to stronger, while maintaining an incremental policy. The author introduces a reliable logical obfuscation technique called random technique (RT) that meets the main requirement of a well-designed logical locking solution.

2.0 Literature Review

With reference to sequential logic obfuscation, an algorithm applicable to digital signal processing circuits such as filters and FFT, employing a high level transformation approach was planned in [5]. This technique used meaningful and non-meaningful modes of design to hide its functionality and render it unusable. A specific application of this technique using an FFT circuit is described in [15], [16]. However, these methods are highly specific to the circuit described and have not been universally demonstrated. This article uses the concept of these schemes to introduce new methods of obfuscation. The proposed dynamic obfuscation scheme leaves the system's data path intact. Modes can be controlled simply by changing the control circuit. [29] proposes a method to protect against side-channel attacks on scan cells included in chips to support testing. Obfuscation applied to scan data was called dynamic obfuscation. However, the dynamic obfuscation technique proposed in [51] and [50] differs from the one proposed in this article. In addition, [31] introduced a technique to protect IP released for evaluation using a hardware Trojan. Our research also uses the same high-level idea of dynamically changing data to protect against attacks. [41] used a hardware Trojan horse circuit to design dynamically obfuscated circuits. However, application of these concepts of conventional hardware obfuscation that protects chips during manufacturing has not yet been covered in previous publications.

Many RE countermeasures, such as circuit-level [4], [47], [27] and [34], and algorithm-level [23], [24] and [13], have been proposed to address RE challenges and prevent IP infringement. At the circuit level, [17], [18] proposed an image-based cloaking method for extracting gate-level netlists by hiding gates [46] or dummy contacts in the layout. Another technique to prevent IP piracy is logical locking [35]. Using XOR/NXOR gates, MUXES, and combinations of these elements, additional encryption blocks are inserted to hide the functionality and implementation of the IC. Therefore, the design will provide accurate functionality only when the correct keys are applied [27]. By combining obfuscated cells with functional circuits that cannot be physically duplicated, licenses can be generated to improve the security of hardware circuits [14]. Unfortunately, on-chip storage of various data is inherently vulnerable to attacks such as SCA, imaging, and error analysis. At the algorithmic level, [13] proposed to protect all states with a cheap state deflection-based obfuscation method that dynamically deflects the state transitions from the original transition path to the black hole cluster if the wrong key is applied. Koteshwara et al. [11] planned a dynamic technique, leading to the modification of obfuscating signals with time. [50] Planned a hologram-based obfuscation, that integrated 2 digital signal process (DSP) kernel architectures during a

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

unseeable manner while not dynamical the practicality of every DSP kernel. However, the introduction of a further code needs a lot of hardware overhead and is additionally susceptible to SCA. Field programmable gate array (FPGA)- primarily based approaches in the main shield device security through algorithms for IoT protection [46], applying algorithms to register transfer level (RTL) codes to understand RTL obfuscation, and in the main defensive against hardware Trojan horses, aspect channel attacks, and then forth [51].

[30] Planned a way that keeps one or a lot of directions encoded (i.e. en-crypted [42], [43] or compressed [44], whereas the program isn't execution and decodes the sequence(s) once the program is running [18] . The resilience of program cryptography against attacks depends on the rule used for cryptography, e.g. a compression rule is undone while not a secret key, whereas associate coding needs finding the key. However, the prices might also be comparatively high compared to different obfuscation techniques, as a result of the code should be decoded before it is dead. There's a trade-off between resilience and value locking on the amount of roughness if transformation is applied [11], that is, if applied at instruction level [2]. The price and resilience measures are high as a result of the decoded instruction and re-encoded, hence the complete code is kept in decoded memory type. [45] at one time proposed a program cryptography, applied at program level, decoded and later starts execution, hence, associated aggressor performed a memory dump and decryption to possess a replica of the complete code. [42] in addition, proposed a system that doesn't shield well against dynamic analysis attacks, throughout execution, the code is decoded in memory and it is changed directly in memory by the MATE aggressor [50].

Logic protection algorithms are developed in response to rising threats against the hardware provide chain; particularly, these techniques are to mitigate the risks of information science piracy through reverse engineering [8], IC overrun, and Trojan insertion [9]. The essence of this approach is to change the planning by adding a protection mechanism, creating it tougher for associate someone to steal style secrets, produce unauthorized copies of invented chips, or perform a purposeful modification for the netlist to insert a Trojan. This section reviews existing logic protection algorithms and connected attacks.

3.0. Research Methodology

3.1.0. Insertion Phase

In this section, we have a tendency to choose the positions where locking key gates are going to be inserted. The choice in this study is formed haphazardly (randomly) and the key gates are inserted in those designated positions. The choice process varies with different proposed methodology. The random methodology enforced for position selection with the obfuscation benchmark suite is represented below. – **Random**: the thought of inserting locking gates in random positions is comparable to the thought of position choice projected in [4]. As represented before, the only methodology of key gates insertion is inserting the key gates haphazardly within the circuit. The algorithm is depicted below:

3.1.1 Algorithm 1: Random Position choice

1: list <Node> RandSel[C:circuity, LN=list<Int>]

2: list <Node > LP={ }

3: list < Modules > LM=extract Modules (c)

4: $\forall M_i \in LM$ *do*

5: $\forall j \leftarrow 1-to-N_i$ *do*

6: Index rand=Random (size, $M_i \in Nodes$

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

7: for $N_{sel} = M_i, Nodes[rand]$

8: while $N_{sel} \in LP$ *do*

9: rand =Random (size ($(M_i, Nodes)$ *)*

10: for $N_{sel} = M_i, Nodes[rand]$

11: LP add (N_{sel})

12: return LP

3.1.2. Identification of Locking Points

When a locking point 'i' has O_i alternatives, the decision can be represented with an integer value between 1 and O. For example, if an addition can be locked with two types of "fake" operations: subtraction and multiplication, the corresponding element can take on: 0 (no locking), 1 (lock with subtraction), and 2 (lock with multiplication). On the contrary, a control branch can assume only two values: 0 (no locking), 1 (locking). So, the analysis creates a vector of integers that represents decisions for all locking points. The integer vector representing a locking solution has as many elements as the number of locking points. The number of key bits Ks required to lock a solution is:

$$K^s = \sum_{1}^{N^c} b_c.B_c + \sum_{1}^{N^a} b_a + \sum_{1}^{N^b} b_b \quad \textbf{(6)}$$

where N^c , N^a and N^b are the total number of locking points for constants, operations and branches, respectively [20] [21].

The random logic locking process [51] locks the circuit by inserting XOR key-gates at random locations during a netlist. Figure 3 shows a netlist latched with 2 key-gates, K1 and K2, mistreatment random logic locking. In random logic locking, the key gates measure unfolds uniformly within the entire netlist [44]. Since the placement is random, it is assumed to be uncorrelated key bits. Considering these assumptions, every bit is delineated as a binary symmetrical channel within a binary wiretap model, as shown in figure 3. In this classical model, the proposed enumerations of the combine key bits, is achieved by setting the input (input a = 1, b = 0 and c = 0). One will use this pattern to work out the worth of G1 in Output 2. These bits are inputs to some extra logic gates inserted into the netlist. Physically unclonable functions (PUFs) are functions derived from physical properties of semiconductor components that are distinctive to every part [46]. The key bits for logic cryptography is derived from PUFs, basically providing a singular key for every chip. Netlist level obfuscation is taken into account as a measure against hardware trojans [13]. The fact that control gates and the associated pact resemble the AND/NAND gates in weighted random pattern generation, we call the proposed approach logic locking.

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071

Volume: 4 Issue: 10 | October 2022

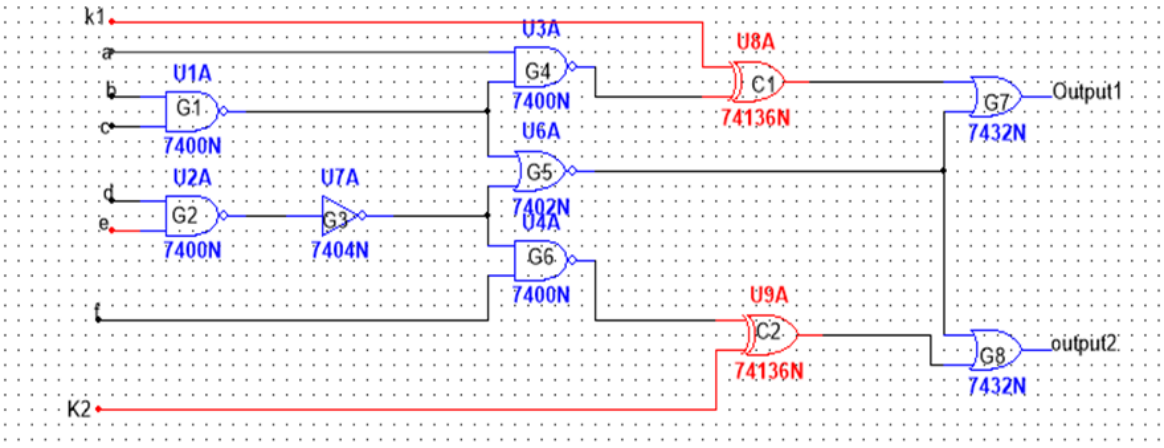

Figure 3: A Circuit with Two Camouflaged Gates C1 and C2 Which Can Be Resolved Individually.

3.1.4. Functionality Strip

The first modification of the initial netlist may be a functionality strip operation, shown in Figure 4. Its final goal is to invert the output of the logic cone for all input patterns that has a Hamming Distance, h, from the private key. The program iterates through all 2key size patterns and checks if it's acting distance from the private key is h. For such patterns, associated AND circuit is inserted. Its inputs are inputs of the logic cone. Outputs of all such created AND gates are fed to a recently inserted OR logic gate. The output of the OR logic gate is logic '1' for those input patterns of a hamming distance, h, from the private key. It's then fed to a recently inserted XOR gate whose input is that of the protected logic cone. The output of the XOR gate is then the initial logic cone output for input patterns with a hamming distance, h, and an inverted logic cone output for input patterns with Hamming distance h from the private key.

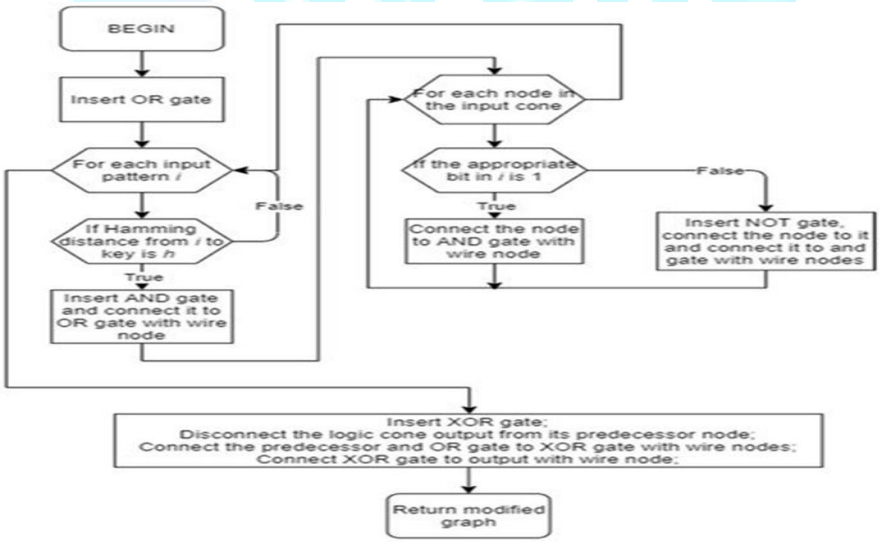

Figure 4: Flowchart for Inserting Logic Locking Gates

3.1.5. Differential Entropy Testbench Template

To estimate the mean differential entropy, the author ran a behavioural simulation of the obfuscated design. That is, a test bench is illustrated to measure differential entropy in our framework with Parameter N_OUT = 256; and parameter DELTA = 0.00000000001;]. This is generated by our framework, and contains inputs keys and golden

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071

Volume: 4 Issue: 10 | October 2022

outputs. A file module is generated with input vectors for both primary input keys, together with outputs obtained by simulating the original design with the same primary input vectors. The testbench for the outputs is similar. The output of the circuit is collected for each combination of primary input and key input to estimate Pi. Then we evaluate the differential entropy with the following formula, previously described in Equation 5:

$$\bar{H} = \frac{1}{N}\sum_{i=1}^{N}\left(p_i \log\frac{1}{1-p_i} + (1-p_i)\log\frac{1}{1-p_i} \right) \quad (7)$$

The testbench designers must know at least the control side of the design. The key points of the testbench are the synchronizing ones. If a design has no control input or output those phases can be omitted but the designers must know the correct cycles in which they should provide the inputs and when to read the corresponding outputs.

The area over head can therefore, be estimated as follows**:**

$$AreaOverhead = \alpha.c + \beta.b + \gamma.a \quad (8)$$

where 'c', 'b', and 'a' are the numbers of bits used for obfuscating constants, $(\alpha, \beta, \&, \gamma)$ are parameters that can be either given by the designer or estimated by the framework. To estimate the overheads parameters, the framework measures the mean percentage overhead for each type of obfuscation point. To do so, it synthesizes and measures the area of the plain design and of three obfuscated designs, each of them obtained by obfuscating all the obfuscation points of the specific category.

3.1.6 To. Assess the Vulnerability of a Locked Circuit

Algorithm 3: Bit –Coloring Attack

Input: The SFLL circuit $c(x,k,y)$, *a protected input pattern* $\hat{x}$, *and original function* $f(x)$

$(x', k') = SAT(restore\_unit(x,k) = 1);$

$h = HD(x', k');$

$c(\hat{x}_0) = red;$

$x' = \hat{x}$ *with* $\hat{x}$ *flipped;*

For $\hat{x}_i \in \hat{x}, 1 \le i \le n-1$ *do*

$\hat{x}'' = \hat{x}'$ *with.*x_i' $----$ *flipped;*

if $(\hat{x}'') = sf(\hat{x}'')$ *then*

$c(\hat{x}_i) = red;$

If else

$c(\hat{x}_i) = green$

$k^* = \hat{x}....with...\hat{x}_i$ *flipped*, $0 \le i \le n-1$, *if* $|c(\hat{x}_i)| = h$

Theorem 3.1.7: Given $\hat{x}, s.t, sf(\hat{x}) \neq f(\hat{x})$, the correct key K* of SFLL can be found with only n-1 queries to an activated IC [48].

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071

Volume: 4 Issue: 10 | October 2022

Proof: we want to color each bit of $\hat{x}$ with green and red colors in a way that:

$$c(\hat{x}_i) = \begin{cases} red, if, \hat{x}_i \neq k_i^* \\ green; otherwise \end{cases} \quad (9)$$

Since K* is unknown, we develop an attack method to find K* by flipping bits of $\hat{x}$. We flip the first bit of $\hat{x}$, then flip a bit $\hat{x}_i$ among the remaining n-1 bits to have $\hat{x}'$ and check whether $f(\hat{x}') \neq sf(\hat{x}')$. Please note that $HD(\hat{x}', \hat{x}) = 2$ (Hamming Distance) and if $f(\hat{x}') \neq sf(\hat{x}'), HD(\hat{x}', K^*) = h.$ Therefore, the flipped bit $\hat{x}_i$ will be colored differently from the first bit $\hat{x}$. we flip $\hat{x}_i$ back, and test the next bit. With these operations, we partitioned the bits of $\hat{x}$ into two groups. Because h is already known, we compare the number of bits in the partitioned groups with h and it is easy to show that at least one of the groups have h number of bits. Therefore, K* can be found by flipping bits of $\hat{x}$ in that group.

Lemma **1**: If the number of bits in two groups are the same after the coloring process illustrated in Theorem 3.5.1, the two choices of a correct key K* are equivalent. [49].

Proof: Such a situation happens when $h = \frac{n}{2}$. We can therefore, find K* or its complement of $\bar{k}^*$ If K* and $\bar{K}^*$ share the same protected input patterns, then K* and $\bar{K}*$ are functionally equivalent. Assuming $\exists \hat{x}, s.t, HD(\hat{x}, K*) = h$ but $HD(\hat{x}, \bar{k}*) \neq h$, since $\bar{k}*$ is the complement of K*, $HD(\hat{x}, \bar{k}*) = n - h.$ Since $h = \frac{n}{2}$, $HD(\hat{x}, k*) = HD(\hat{x}, \bar{k}*) = \frac{n}{2}$ which is a conflict with the assumption. Here is an example for illustration: Assuming K*=10110, h=2 and an attacker has a protected input pattern $\hat{x} = 00010$. It is required to assess the vulnerability of the locked circuit.

Step 1: we flip the first bit of $\hat{x}$ to get 10010

Step 2: we them flip individual bits from $\hat{x}_1, to, \hat{x}_4$.

Step 3: display the coloring result as shown in Table 1.

Table 1: Coloring Result

Flip	$\hat{x}'$	$sf(\hat{x}') = f(\hat{x}')?$	$c(\hat{x}_i)$
$\hat{x}_1$	10010	No	$\neq c(\hat{x}_o)$
$\hat{x}_2$	11010	Yes	$= c(\hat{x}_o)$
$\hat{x}_3$	10110	No	$\neq c(\hat{x}_o)$
$\hat{x}_4$	10000	No	$\neq c(\hat{x}_o)$
$\hat{x}_5$	100110	No	$\neq c(\hat{x}_o)$

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

Step4: since h = 2, we check the group that has the same color and the groups with two bits $(\hat{x}_0 \& \hat{x}_2)$ has the same color.

Step 5: Decision, therefore, k* can be simply found by flipping $(\hat{x}_0 \& \hat{x}_2)$.

3.1.8. Incorporating Logic Locking Attacks into the Random Technique Framework

Understanding information leakage and the effects on security of logic locking is illustrated in figure 5, through the use of a generic circuit. The generic circuit uses an XOR based locking scheme and the placement of key gates is done randomly. Furthermore, since the placement is random, RT is also assumed to have uncorrelated key bits. Considering these assumptions, each bit can be represented as a binary symmetric channel inside a binary wiretap model shown in figure 4.

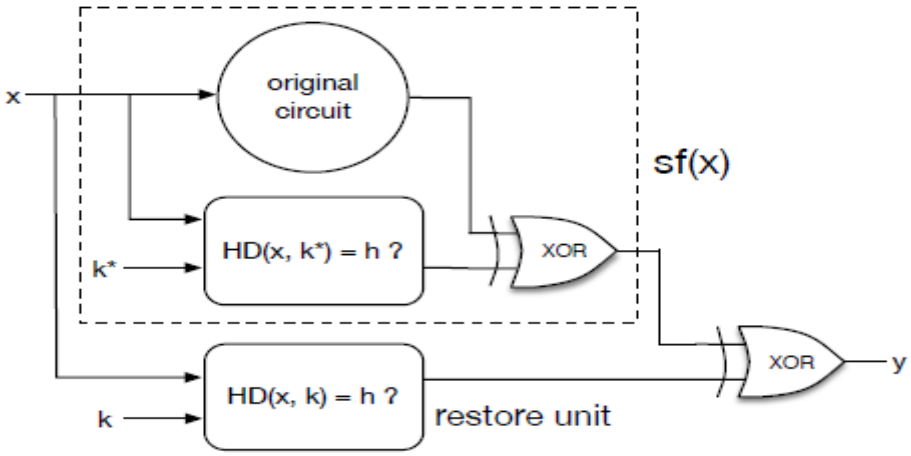

Figure 5: Stripped function logic locking

3.1.9. Result and Discussion

Algorithm 3 is used to explain the logic locked vulnerability and how it affects the security of logic locking. The gate keys are placed at random and the algorithm uses an XOR-based locking technique. In order to tackle random-based logic locking (RLL) (Chen et al., 2021), testability and fault analysis qualities are examined. RLL is a random placement approach used for inserting key-based XORes. This made it possible to compare the obfuscated circuit to its golden source to see whether sufficient structural changes had been made to manage the functionality of the original circuit. Additionally, it gives an estimate of how much work would be necessary for an attacker to defeat the inserted Logic lock.

The distance formula, which determines the difference between two multi-element descriptions, is used to measure this type of distance. Since the value of K^* is unknown, we create an attack strategy that uses bit-flipping of $\hat{x}$ to find K*. We flip the first bit of $\hat{x}', \forall i = 1,...,n$ and determine whether $f(\hat{x}') \neq sf(\hat{x}')$ while noting the Hamming Distance "h." The two options for a proper K* are comparable if the total amount of bits in the two groups after flipping is the same. Therefore, flipping is used to determine Kvalue. *'s To make it difficult for an attacker to determine the proper key by examining the locked circuit is the challenge of logic locking.

3.2.0. Evaluation metric

Matthews Correlation Coefficient (MCC) [7] is employed as a statistic for accuracy due to the imbalances in the data. When there are class inequalities, accuracy alone can deceive the viewer. The MCC formula is displayed in

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071

Volume: 4 Issue: 10 | October 2022

equation (10). MCC considers all components of the confusion matrix, whereas accuracy just considers the true positives and true negatives. It produces a number between -1 and 1, with a higher score indicating a more effective model.

$$MCC = \frac{TPxTN - FPxFN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \qquad (10)$$

Where, TP means True Positive value

TN means True negative value

FP denotes False Positive

FN denotes False Negative value

The Matthews Correlation Coefficients (MCC) describes the following:

"A correlation of:

C = 1 denotes perfect correlation,

C = 0 describes the expected random value, and

C = -1 denotes total disagreement of prediction and observation"`.

For example, four hundred automobile owners were asked to categorize the main reason for selecting the particular make and model of their present car. The two possible categories were (i) performance and (ii) appearance. This is depicted in (Table 2):

Table 2: The computation of the Matthews Correlation Coefficients (MCC)

Sex of Respondents	Responses	
	Appearance	Performance
Male	95 (True Positive)	55 (False Negative)
Female	85 (False Positive)	165 (True Negative)

To calculate the MCC of the model, we can use the following formula of equation (10)

$$MCC = \frac{(95X165)-(85X55)}{\sqrt{(95+85)(85+55)(165+85)(165+55)}}$$

$$= 0.297$$

0.297 denotes a classifier that is close to a random guess classifier (0).

The code below shows array of class prediction and the computation of Matthews correlation coefficient of a model in Python:

import numpy as np

INTERNATIONAL JOURNAL ON ORANGE TECHNOLOGY

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071

Volume: 4 Issue: 10 | October 2022

from sklearn.metrics import matthews_corrcoef

#define array of actual classes

actual = np.repeat([1, 0], repeats=[165, 235])#define array of predicted classes

pred = np.repeat([1, 0, 1, 0], repeats=[95,85,55, 165])

#calculate Matthews correlation coefficient

matthews_corrcoef(actual, pred)

0.2971421052631579

This confirms the manually computed value..

3.2.1. Computation of Balanced Accuracy

This is a metric to assess the performance of a classifier model.

It is computed as follows:

$$Consistency(balanced).accuracy = \frac{sensitivity}{2} \quad (11)$$

Where:

The percentage of positive cases the model is able to identify is called the "true positive rate," or sensitivity.

The percentage of negative cases that the model is able to identify is known as the "real negative rate" or specificity.

This statistic is helpful for categorizing cases with imbalances. The model's predictions are enumerated in the confusion matrix below (see table 3)

Table 3: Summarizes the prediction model

Sex of Respondents	Responses	
	Appearance	Performance
Male	95 (True Positive)	55 (False Negative)
Female	85 (False Positive)	165 (True Negative)

Therefore, to compute the balanced accuracy of the model:

i. first calculate the sensitivity and specificity:

Sensitivity: The “true positive rate” $= \frac{TP}{(TP+FP)} = \frac{95}{(95+85)} = 0.5277$

Specificity: The “true negative rate” $= \frac{TN}{(TN+FN)} = \frac{165}{165+55} = 0.75$

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

ii. Then compute the balanced accuracy as:

Balanced accuracy score = (Sensitivity + Specificity) / 2

$$\text{Balanced accuracy score} = \frac{(0.5277 + 0.75)}{2} = 0.63885$$

Therefore, the balanced accuracy for the model is (0.63885).

The Python code below demonstrates how to define an array of predicted classes and an array of actual classes, as well as how to compute the model's balanced accuracy:

3.2.2. Example: Calculating Balanced Accuracy in Python

```
import numpy as np
from sklearn.metrics import balanced_accuracy_score
#define array of actual classes
actual = np.repeat([1, 0], repeats=[165, 235])
#define array of predicted classes
pred = np.repeat([1, 0, 1, 0], repeats=[95, 85, 55, 165])
#calculate balanced accuracy score
balanced_accuracy_score(actual, pred)
```

0.63885 This confirms the manually computed value.

3.2.3. Conclusion

Logic locking has emerged as an obfuscation technique to protect out sourced chip designs, where the circuit netlist is locked and can only be functional when a secure key is programmed. The Boolean satisfiability-based attacks show how to break logic locking circuit, and simultaneously motivate researchers to develop more secure countermeasures. In this paper, we present a novel fault injection attack to break any locking technique that relies on a stored secret key, and denote this technique as Random Technique (RT). To evaluate the logic encryption robustness, two major criteria were utilized, which are (i) the interdependency between the keys and (ii) the output corruption against attacks, including path sensitization attack. Algorithm 2, is used to assess the vulnerability of the locked circuit. This allowed the obfuscated circuit to be measured against its golden source to indicate if enough structural change has been implemented to handle its functionality. The RT of this study, developed an attack method to find the value of the secret key, K*, which is unknown. This study carefully analyzed the security of SFLL, and suggested a smart method to find the hamming distance, ’h’ by one or more SAT query. It utilized the regularity between the protected input patterns and the secret key to propose an efficient bit-coloring attack. The attack deciphers the secret key, K*, with (n-1) queries by flipping bits. The result shows that given a protected input pattern, the correct key can be found in a very short while by flipping bits. This article, also shows how MCC produces a more informative and truthful score in evaluating binary classifications for accuracy.

The proposed reconvergence metric is found to be proportional to key sensitization attack resiliency and inversely proportional to SAT attack resiliency. These conclusions will motivate future work to improve the proposed metrics, develop new ones, and utilize the metrics in security assessment.

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

3.2.4. Acknowledgement

I would like to express my special appreciation and thanks to my advisor professor Akinwale A.T of Federal University of Agriculture Abeokuta Nigeria, you have been a tremendous mentor for me. I wish to use this medium to express my thanks to you for encouraging my research and allowing me to grow as a research scientist. Your advice on both research as well as on my career have been invaluable.

Nobody has been more important to me in the pursuit of this article than the members of my family. Most importantly, I would like to thank my loving and supportive wife, Evangelist Martha, and my four wonderful children, Pamela, Ohi, Ebehitale and Ehis, who provide unending inspirations.

3.2.5. Refernces

1. **A. Saha, D. Mukhopadhyay and R. S. Chakraborty (2021):** "Design and Analysis of Logic Locking Techniques," *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC), 2021, pp. 1-2, https://doi.org/10.1109/VLSI-SoC53125.2021.9606975.*
2. **Alan Rodrigo Diaz RizoJulian LeonhardHassan AboushadyHassan AboushadyHaralampos-G. Stratigopoulos (2022):** "RF Transceiver Security Against Piracy Attacks January 2022." Circuits and Systems II: Express Briefs, IEEE Transactions on https://doi.org/10.1109/TCSii.2022.3165709
3. **Animesh Chhotaray and Thomas Shrimpton (2022):** "Hardening Circuit-Design IP Against Reverse-Engineering Attacks". *2022 IEEE Symposium on Security and Privacy (SP) Year: 2022, Pages: 379-396. https://doi.org/10.1109/SP46214.2022.00023.*
4. **Antonios Pavlidis, Eric Faehn, Eric Faehn, Marie-Minerve Louerat, Marie-Minerve Louerat, Haralampos-G. Stratigopoulos (2022)**; "Run-Time Hardware Trojan Detection in Analog and Mixed-Signal ICs April 2022 Conference: 2022 IEEE 40th VLSI Test Symposium (VTS). https://doi.org/10.1109/VTS52500.2021.9794208.
5. **Bellizia, D., Scotti, G., Trifiletti, A. (2018): "**logic style as a countermeasure against side-channel attacks: secure cells and experimental results". *IEEE Trans. Circuits Syst. I: Reg. Papers. 65(11), 3874–3884 (2018).*
6. **Chatterjee, P., Chatterjee, A., Campos, J., Abreu, R., Roy, S.(2020):** "Diagnosing software faults using multiverse analysis." In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. pp. 1629–1635. International Joint Conferences on Artificial Intelligence Organization 2020). https://doi.org/10.24963/ijcai.2020/226, main track. https://doi.org/1024963/ijcai.3030/226.
7. **Chicco, D., Tötsch, N. & Jurman, G (2021*):*** The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Mining 14, 13 (2021). https://doi.org/10.1186/s13040-021-00244-z*
8. **Christoph Kerschbaumer Tom Ritter, Frederik Braun (2020):** "Hardening Firefox against Injection Attacks".*2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)***,** *Year: 2020, Pages: 653-663***.**
9. **Dongpeng Xu, Binibin Liu, Weijii Feng. Jiang Ming, Oilong Zheng, Jing li, Olaoyan Yu (2021);** "Boosting SMT Solver Performance on Mixed Bitwise – arithmetic expressions. PLDI 2021" : Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, june 2021, pages 651 – 664. https://doi.org/10.1145/3453483.3454068

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

10. **Golia, P., Roy, S., Meel, K.S**.(2021): "Program synthesis as dependency quantified formula modulo theory." In: Zhou, Z.H. (ed.) Proceedings of the Thirtieth In- ternational Joint Conference on Artificial Intelligence, IJCAI-21. pp. 1894– 1900. International Joint Conferences on Artificial Intelligence Organization (2021). https://doi.org/10.24963/ijcai.2021/261.

11. **Gomez, H., Duran, C., Roa, E. (2019):** Defeating silicon reverse engineering using a layout- level standard cell camouflage. *IEEE Trans. Consum. Electron. 65(1), 109- 118 (2019).*

12. **Hadeel Alrubayyi, Gokop Geteng, Mona Jaber and James Kell (2021**): "Challenges of Malware Detection in the IoT and Review of Artificial Immune System, Approaches. Sensor and Actuator Networks. Journal of Sensor Actuator Network _ 2021, 10(4), 61; https://doi.org/10.3390/jsan 10040061.

https://doi.org/10.1109/EuroSPW51379.2020.00094.

13. **Isyak MeirobieAgustinus Purna IrawanAgustinus Purna IrawanHusni Teja SukmanaHusni Teja Sukmana, Nuke Puji Lestari SantosoNuke Puji Lestari Santoso (2022**): *"*Framework Authentication e-document using Block chain Technology on the Government system". *July 2022 International Journal of Artificial Intelligence Research 6(2) https://doi.org/10.29099/ijair.v6i2.294.*

14. **Jean Paul A., Yeacoub, Ola-salman, Hassan N. Moura, Nesirene Kaaniche, Ali Chehab and Mohammad Malli (2020):** "Cyber – Physical Systems security; Limitations, Issues and Future trends. Microprocessor Microsystem 2020, 77, 103201, https://doi.org/10.1016/micpro.2020.103201.

15. **Jianqi Chen and Benjamin Carrion Schafer (2021),** "Area Efficient Functional Locking through Coarse Grained Runtime Reconfigurable Architectures," 2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC), 2021, pp. 542-547.

16. **Kyle Juretus and Ioannis Savidis (2021**): "Increased Output Corruption and Structural Attack Resilience for SAT Attack Secure Logic Locking". IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 40, NO. 1, JANUARY 2021, pp.38-51. https://doi.org/*10.1109/TCAD.2020.2988629*

17. **Leonidas Lavdas, M Tanjidur Rahman, Mark Tehranipoor, Navid Asadizanjani (2020**), "On Optical Attacks Making Logic Obfuscation Fragile", 2020 IEEE International Test Conference in Asia (ITC-Asia), pp.71-76, 2020.

18. **Leonidas Lavdas, M. Tanjidur Rahman, Navid Asadizanjani (2021**), "Application of Optical Techniques to Hardware Assurance", Emerging Topics in Hardware Security, pp.471, 2021.

19. **Lilas Alrahis, Satwik Patnaik, Johann Knechtel, Hani Saleh, Senior, Baker Mohammad, Mahmoud Al-Qutayri, and Ozgur Sinanoglu,(2021**): "UNSAIL: Thwarting Oracle-Less Machine Learning Attacks on Logic Locking." Article in IEEE Transactions on Information Forensics and Security· February 2021. https://doi.org/101109/TIFS.2021.3057576.

20. **Limaye, N., Patnaik, S., & Sinanoglu, O. (2022).** Valkyrie: Vulnerability Assessment Tool and Attack for Provably- Secure Logic Locking Techniques. *IEEE Transactions on Information Forensics and Security, 17, 744-759. https://doi.org/10.1109/TIFS.2022.3149147.*

21. **Lucas Barthelemy, Ninon Eyrolles, Guenael Renault, Raphael Roblin (2016**): " Binary Permutation Polynomial Inversion and Applications to Obfuscation Techniques". SPRO'16': Proceedings of the 2016 ACM workshop on Software Protection (2016). Pages 51-59. https://doi.org/10.1145/2995306.2995310.

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

22. **[M. Tanjdur Rahman; M. Sazadur Rahman; Haunuu Wang; Shir Tajik; Waled Khalil; Falamah Faramah; Domenic Forte; Navid Asadizanjarin; Mark Tehranipoor** (2020): " Defence-in-depth: A receipt for Logic Locking to Prevail. Integration, the VLSI Journal 72 (2020), 39-57. https://doi.org/10.1016/j.vlsi.2019.12.007.

23. **M. Weiner, S. Manich, R. Rodrıguez-Monta~nes, G. Sigl (2018):** "The low area probing detector as a countermeasure against invasive attacks, IEEE Trans. Very Large Scale Integr. Syst. 26 (2) (2018) 392–403, https://doi.org/10.1109/TVLSI.2017.2762630.

24. **M. Zuzak, A. Mondal and A. Srivastava (2022)***: "*Evaluating the Security of Logic- Locked Probabilistic Circuits," *in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.41, no.7, pp.2004-2009, July2022, https://doi.org/10.1109/TCAD.2021.3104270.*

25. **Mengnan Chen, Yongquan Zhou and Qifang Luo(2022***): "*An Improved Arithmetic Optimization Algorithm for Numerical Optimization Problems". *Mathematics 2022, 10(12), 2152; https://doi.org/10.3390/math10122152 - 20 Jun 2022*

26. **Mohamed tarek, ahmed Elshamy, Alhassan Sayed, Marie -Minerve Louerat [...] Haralampos-G. Stratigopoulos**. (2021): "Locking by Untuning: A Lock-Less Approach for Analog and Mixed-Signal IC Security. Article: November 2021IEEE Transactions on Very Large Scale Integration (VLSI). November 2021IEEE Transactions on Very Large Scale Integration (VLSI) Systems PP(99). https://doi.org/10.1109/TVLSI.2021.3117584.

27. **Pallari ahire, Jibi Abraham (2022**): "Secure Cloud Model for Intellectual Privacy Protection of Arithmetic Expressions in Source Codes Using Data Obfuscation techniques. Journal of Theoretical Computer Science, Volume 922, 24th June 2022, pages 131-149. https://doi.org/101016/j.tcs.2022.04.018.

28. **Robin David, Luigi Coniglio, Mariano Ceccato (2020**); " QSynth-A Program Syunthesis Based Approach for Binary code Deobfuscation workshop on Binary Analysis. *Research (BAR) 2020 ISBN-891562-62-2. https://doi.org/10.14722/bar.2020.23009*

29. **Roy, S., Hsu, J., Albarghouthi** (2021), A.: Learning differentially private mechanisms. In: 2021 2021 IEEE Symposium on Security and Privacy (SP). pp.852–865. IEEE Computer Society, Los Alamitos, CA, USA (May 2021). https://doi.org/10.1109/SP40001.2021.00060

30. **Roy, S., Pandey, A., Dolan-Gavitt, B., Hu, Y**.(2018): "Bug synthesis: Challenging bug- finding tools with deep faults." In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 224–234. ESEC/FSE 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3236024.3236084

31. **Ruijie Meng, Biyun Zhu, Hao YunHaicheng Li, Yan Cai, Zijiang Yang,(2019**): "An Effective Tool for Detecting Concurrency Vulnerabilities". 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) Year: 2019, Pages: 1154-1157 https://doi.org/10.1109/ASE.2019.00125

32. **Sarah Amir; Bicky Shaka; Xiadin Xu; Yier Jin; Swarup Bhunia;Mark Tehranipoor; Domenic Forte (2018**): "Development and Evaluation of Hardware Obfuscation Benchmarks." *Journal of Hardware and Systems Security (2018) 2:142-161*. https://doi.org/10.1007/s41635-018-0036-3.

33. **Sebastian banescu, Alexander Pretschner (2018**): "A Tutorial on Software Obfuscation. Advances in computers, Volume 108, 2018 page 283 – 353 https://doi.org/10.1016/bs.adcom.2017.09.004.

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

34. **Shamsi, K., Li, M., Plaks, K., Fazzari, S., Pan, D.Z., Jin, Y.(2019**): "IP-Protection and Supply Chain Security through Logic Obfuscation: A Systematic Overview. Trans. on Design Automation of Electronic Systems (TODAES) p. 65 (2019)

35. **Shamsi, K., Pan, D.Z., Jin, Y.(2019**): "On the Impossibility of Approximation-Resilient Circuit Locking". In: 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 161–170. IEEE (2019)

36. **Shuai Jang; Yao Hong, Cai Fu, yekui Qian, Lansheng Han (2021**): "Function-Level Obfuscation detection method based on Graph Convolutional Networks." Journal of Information security and Applications, Volume 61, September 2021, 102953. https://doi.org/10.1016/j.jisa.2021.102953.

37. **Singal, D., Agarwal, P., Jhunjhunwala, S., Roy, S**.(2018): Parse condition: Symbolic encoding of ll(1) parsing. In: Barthe, G., Sutcliffe, G., Veanes, M.(eds.) LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning. EPiC Series in Computing, vol. 57, pp. 637–655. Easy Chair (2018). https://doi.org/10.29007/2ndp

38. **Shrone, D., Subramanyan, P.**(2020): "Functional Analysis Attacks on Logic Locking. IEEE Transactions on Information Forensics and Security 15, 2514–2527 (2020)

39. **Sisejkovic, D., Merchant, F., Reimann, L.M., Leupers, R.(2021):** "Deceptive logic locking for hardware integrity protection against machine learning attacks." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2021)

40. **Sungkwang Lee, Nam-su jho, Doyoung Chung, Yousung Kang, Myungchul Kim (2022**): " RCRYPTECT: Real-time Detection of Cryptographic function in the User-Space Filesystem". Journal of Computers and Security, Volume 112, January 2022, 102512. https://doi.org/10.1016/jcose.2021.102512

41. **T. Hoque, R. S. Chakraborty and S. Bhunia (2020**), "Hardware Obfuscation and Logic Locking: A Tutorial Introduction," *in IEEE Design & Test, vol. 37, no. 3, pp. 59-77, June 2020, https://doi.org/10.1109/MDAT.2020.2984224.*

42. **Tamzidul Hoque, Raja Subhra Chakraborty and Swarup Bhunia (2020**), "Hardware Obfuscation and Logic Locking: A Tutorial Introduction," in IEEE Design & Test, vol. 37, no. 3, pp. 59-77, June 2020; https://doi.org/10.1109/MDAT.2020.2984224

43. **V. V. Rao, K. Juretus and I. Savidis (2020)***: "*Security Vulnerabilities of Obfuscated Analog Circuits*," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020, pp. 1-5, https://doi.org/10.1109/ISCAS45731.2020.9180781.*

44. **Verma, A., Kalita, P.K., Pandey, A., Roy, S.(2020**): "Interactive debugging of con- current programs under relaxed memory models". In: Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization. p. 68–80. CGO 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3368826.3377910

45. **Weijii Feng, Binbin Lui, Yun Xu, Dongpeng Xu, Oilong Zheng (2020**): " Neureduce: reducing Mixed Boolean Arithmetic Expressions by Recurrent Neural Network. In findings of Association for Computational Linguistics: EMNLP2020, pages 635-644. https://doi.org/10.18653/vi/202.findings=emnip.56

46. **Xianmiao Zhang and Yingjie Lao (2019**), "On the Construction of Composite Finite Fields for Hardware Obfuscation" in IEEE Transactions on Computers, vol. 68, no. 09, pp. 1353-1364, 2019. https://doi.org/10.1109/TC.2019.2901483.

https://journals.researchparks.org/index.php/IJOT e-ISSN: 2615-8140 | p-ISSN: 2615-7071
Volume: 4 Issue: 10 | October 2022

47. **Y. Zhang, A. Jain, P. Cui, Z. Zhou, and U. Guin (2021):** "A Novel Topology-guided Attack and its Countermeasure towards Secure Logic Locking," Journal of Cryptographic Engineering, vol. 11, no. 3, pp. 213–226, 2021.

48. **Yadi Zhong, Yadi Zhong, Ujjwal Guin, Ujjwal Guin** (2022): "AFIA: ATPG-Guided Fault Injection Attack on Secure Logic Locking". June 2022 Cryptography and Security (cs.CR). https://doi.org/10.48550/arXiv.2206.04754.

49. **Yasin, M., Mazumdar, B., Sinanoglu, O., Rajendran, J**. (2020): "Removal attacks on logic locking and camouflaging techniques." IEEE Transactions on Emerging Topics in Computing 8(2), 517–532 (2020)

50. **Yasin, M., Rajendran, J., & Sinanoglu, O**. *(2020). "*The Need for Logic Locking*." In Analog Circuits and Signal Processing (pp. 1-16). (Analog Circuits and Signal Processing). Springer. https://doi.org/10.1007/978-3-030-15334-2_1*.

51. **Yuejun Zhang; Qiufeng Wu; Pengjun Wang; Liang Wen; Zhicun Luan; Chongtan Gu (2021)"**TVD-PB Logic Based on Camouflaging Circuit for IoT Security." Institution of Engineering Technology Circuits, Devices and Systems, Volume 16, Issue 1, pp. 40-52; https://doi.org/10.1049/cds2.12080.

52. **Zhang, H., Yang, W., Fedyukovich, G., Gupta, A., Malik, S**.(2020): "Synthesizing environment invariants for modular hardware verification. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11990 LNCS, 202–225 (2020). https://doi.org/10.1007/978-3-030-39322-9 10

53. **Zhanhao Chen, Yinzhi Cao (2020): "**Fortifying JavaScript against Web Concurrency Attacks via a Kernel-Like Structure". 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) JSKernel: Year: 2020, Pages: 64-75. https://doi.org/10.1109/DSN48063.2020.00026.