2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)

Linear Approximation and Differential Attacks on Logic Locking Techniques

Ghanshyam Bairwa*, Souvik Mandal*, Tatavarthy Venkat Nikhil* and Bodhisatwa Mazumdar*
*Discipline of Computer Science and Engineering
Indian Institute of Technology Indore,
Indore, India.

***Abstract*—Logic locking is a protection technique for outsourced integrated circuit (IC) designs that thwarts IC piracy and IC counterfeiting by untrusted foundries. In logic locking, the key gates used for masking the circuit adds to the area overhead of the original circuit, and hence increases the power consumption too. In this paper, we mount linear approximation attacks and differential attacks on random logic locking (RLL), fault-analysis based logic locking (FLL), and strong logic locking (SLL) techniques. We present our results on ISCAS'85 benchmark circuits. In linear approximation attack, the combinatorial blocks partitioned and expressed as linear expressions to derive a relation between the key inputs and the primary inputs of the circuit. In differential attacks, we could recover the embedded secret key in device with attack effort lesser than exhaustive search attack.**

***Index Terms*—Logic locking, key gate, linear approximation, input differential, output differential.**

I. Introduction

As the complexity of constructing and/or maintaining a fabrication facility with advanced capabilities increases exponentially, numerous semiconductor companies are going fabless. The fabless companies design integrated circuits (ICs) that are manufactured by an off-shore fabrication industry. This outsourcing of IC fabrication enables the design companies to access advanced semiconductor technology at a lower cost. However, outsourcing leads to security threats as the offshore foundry may not be trustworthy [2]. Such untrusted elements in the foundry lead to multiple security threats in the IC manufacturing supply chain that comprise overproduction, Trojan insertion, reverse engineering, intellectual property (IP) theft, and counterfeiting [3]. As such threats lead to annual losses in millions of dollars to the semiconductor industry, hardware security has become imperative in present day scenario [4].

To thwart adversaries from mounting such attacks, a number of hardware design-for-trust (DfTr) techniques have been proposed. Of all these protection techniques, logic locking, gained significant interest owing to its versatility of protection from an attacker at any point of IC supply chain. It comprises obfuscation methods to hide and/or lock the functionality of a netlist have been proposed. Fig 1 demonstrates different phases of the IC design flow; the fabrication industry, the testing package assembly, and the IC activation entity are the untrusted elements of the IC design flow. *Logic locking* is an IC protection technique that hides the functionality and implementation of a circuit design by adding additional gates into the design, thereby thwarting reverse engineering and overproduction threats. The gates inserted in the design are called *key gates*. A key gate can be an XOR/XNOR gate [2], [4], [5], [6], a multiplexer [7], or a look-up table (LUT) [8]. One input of the key gate is an intermediate output of the original design while the other input is the key input. The key inputs are driven from an on-chip memory. To exhibit correct functionality of the design, a valid key needs to be set in the on-chip memory. A tamper-proof chip protection is implemented to prevent untrusted foundries from probing on the internal wires of the design. The locking hardware with key gates renders the IC unusable if correct key is not applied into the key inputs.

The earliest attempt of logic locking called EPIC (Ending Piracy of ICs) provided a complete logic locking framework [2]. However, it suffered from a small Hamming distance between the correct outputs and incorrect outputs corresponding to the same set of inputs. Other logic locking techniques focused on determining the best locations for inserting key gates, such as random [2], fault-analysis based [6], and strong interference based logic locking [9]. Such proposed techniques were subsequently followed by key recovery attacks that exploited vulnerabilities of logic locking techniques, such as satisfiability-based (SAT) attack [10], sensitization [9], AppSAT [11], and signal probability skew (SPS) attacks [12]. The recent works in this area, such as, Anti-SAT [8], TTLock [13], and stripped functionality based logic locking (SFLL) [14], are primarily focused on thwarting SAT and removal attacks.

In this paper, we present two attacks, namely, linear approximation attack and differential attack on RLL, FLL, and SLL circuits. In linear approximation attack, we first compute the linear approximation of the known logic block that results in minimal corruption of output bits. For the entire circuit, this results in multiple linear approximation circuits along with the gates of the circuit. This yields relations between the key gates; the relations are validated with the set of correct key gate tuple values. Further, we mounted differential attack on the three logic locking techniques in which we propagate a constrained differential from the inputs and monitor its propagation through a set of key gates in its path. We identified propagation of multiple differentials within the logic locked circuits that were constrained with circuit partitioning algorithms.

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00081

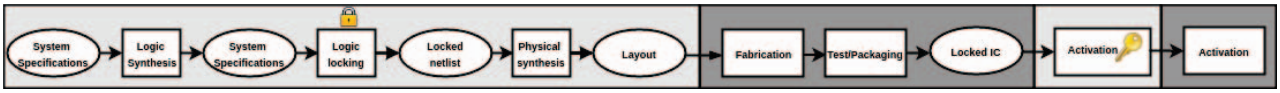


Figure 1. Logic locking in IC design flow. The dark gray regions indicate untrusted phases in the design whereas the light gray regions denote trusted entities. [1]

The paper is organized as follows. Section II illustrates the background on present logic locking algorithms, attacks, and their respective defenses in the existing literature. Section III presents an insight into the proposed linear approximation attack on logic locked circuits. Section IV focuses on the differential attack on the logic locked circuits. Section V discusses the experiments and the corresponding results on the logic locked version of ISCAS'85 benchmark circuits. Section VI concludes the paper.

II. Background

The original circuit netlist is a vectored Boolean function $F : \{0,1\}^n \mapsto \{0,1\}^m$ comprising n inputs and m outputs, referred to as sets $I = \{0,1\}^n$ and $O = \{0,1\}^m$. For each output O_i, $0 \leq i \leq (m-1)$, a *logic cone* defines the circuit that yields O_i. The logic cone is defined as, $LC^i : \{0,1\}^{n_1} \mapsto \{0,1\}$, which comprises of n_1 inputs. The locked netlist of F is denoted as, $F_L : \{0,1\} \times \{0,1\}^k \mapsto \{0,1\}^n$, that comprises k key gates referred to as $K = \{0,1\}^k$ in addition to the n inputs and m outputs. If F_L is activated with the correct key, k_c, then $\forall i \in \{0,1\}^n$, $F_L(i, k_c) = F(i)$. For other key inputs, k_{inc}, $\exists i \in I$, such that $F_L(i, k_{inc}) \neq F(i)$. Similarly, a locked logic cone is defined as, $LC_L^i : \{0,1\}^{n_1} \times \{0,1\}^{k_1} \mapsto \{0,1\}$, i.e., the logic cone LC comprises of k_1 key gates in its locked netlist.

The traditional logic locking techniques that exist in literature are as follows:

1) **Random logic locking (RLL)**: This technique inserts key gates at random locations in a netlist. In RLL circuit, the interference between the key gates are minimal, thus leading to attack vulnerabilities. RLL has been shown to be vulnerable to attacks such as SAT attack that exploit the algorithmic weakness of this logic locking technique.
2) **Fault-analysis based logic locking (FLL)**: This technique prevents black box usage of an IC while mounting an attack, thus the attacker needs to determine the circuit implementation while mounting an attack. The technique maximizes I_{inc} where $I_{inc} = \{i \in I : L_F(i, k_{inc}) \neq F(i)\}$. The key gates are inserted at those nodes of the circuit, which maximize $|I_{inc}|$ when incorrect key values are applied. For FLL implementation, some algorithms employ VLSI test based algorithms to maximize output corruption, whereas other logic locking methods use graph centrality indicators with smaller computational effort.
3) **Strong logic locking (SLL)**: In sensitization attack on FLL, the attacker aims to sensitize each individual key gate to the logic cone output. In SLL, the designers claim that the circuit topology involving key gates cannot sensitize any key to the output due to the mutual interference of at least two key gates a logic cone output. Due to the mutual interference between the keys, an attacker is forced to resolve multiple keys simultaneously, rather than a single key.

In the proposed attack as well as all existing attacks on these logic locked circuits, the threat model involves the following aspects:

(i) the circuit designer and the design tools are trusted,
(ii) the foundry, the test-facility, and the end user of an IC are untrusted,
(iii) a linear approximation (LA) attacker and differential analysis (DA) attacker has access to both the locked netlist and the functionally activated IC. The attacker also has the knowledge of the logic locking algorithm used and the location of the key gates in locked circuit. The unknown parameter is the key value, which is a binary vector.

III. Proposed Linear Approximation Attack on LL circuits

In linear approximation attack, the logic locked circuit F_L is first partitioned into locked logic cones LC_L^i that comprises key gates by employing the depth first search (DFS) algorithm on the entire circuit, where i is the number of output bits in the circuit. Each LC_L^i contains all inputs that determine the output of the logic cone; the LC_L^i is then partitioned into small logic blocks as shown in Algorithm 1. The partitioning of the circuit is based on the location of key gates in the logic cone. The partitioning of the circuit is followed by linear approximation of each of the logic blocks. Subsequently, the linear approximation expressions are combined to yield an approximation involving the primary inputs, primary outputs, key inputs and some intermediate signals of the logic blocks. Henceforth, we denote $I \rightsquigarrow Y$ if signal I drives or sensitizes signal Y, and $I \not\rightsquigarrow Y$ if signal I fails to sensitize signal Y.

1) Algorithm for circuit partitioning: : In circuit partitioning shown in Algorithm 1, the entire circuit is first partitioned into logic cones LC_L^i, $(1 \leq i \leq n)$; each logic cone comprises a combinatorial output O_i, the set of inputs that drive O_i, and the combinatorial block that computes O_i. Each LC_L^i is then further partitioned into logic blocks LB_L^j such that each logic block comprises at least one key gate from the set KG. The partitioning start from the output of the logic cone and proceeds towards the primary inputs. In Figure 3, LB_L^1 is created, which is followed by construction of LB_L^2 and LB_L^3, respectively. Each logic block LB_L^j comprises a set of gates or vertices (PS_V^j), key gates (KG_V^j), and set of wires between gates or edges PS_E^j. As partitioning algorithm constructs a

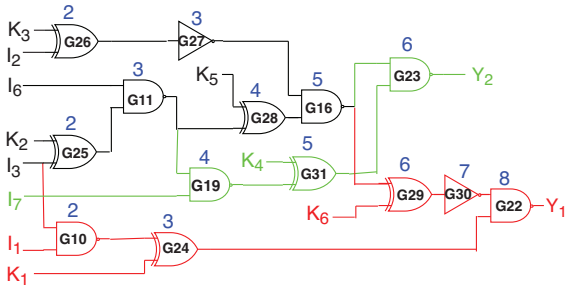


Figure 2. Linear Approximation attack on logic cones of a circuit. Integers on key gates in blue color indicate the height of the gate from the input; the height of a primary input or a key input is 1. The red color and green color parts of the circuit correspond to two separate partitions of the circuit, while the black part belongs to both partitions of the circuit.

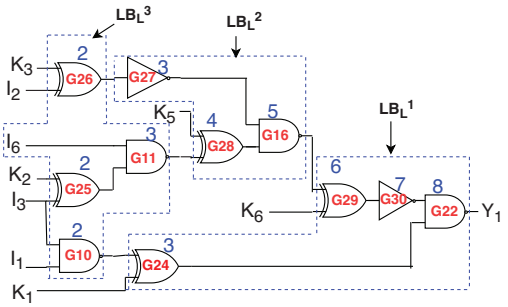


Figure 3. Each partition of a circuit is separated by dashed boundary. The partitions are labelled in green circles. The height of each logic gate is mentioned in a blue color.

logic block, it constructs three sets, PS_V^j, KG_V^j, and PS_E^j; the process continues until all gates have been considered for partition. The partitioning algorithm terminates when a key gate has been considered in the partition. Hence a partition starts with either a logic gate driving a primary output, or a logic gate driving a key gate. For instance, in Figure 3, the logic block LB_L^2 comprises gate $G16$ when the construction of the partition starts; construction of LB_L^2 finishes when the key gate $G28$ is encountered by the algorithm.

2) Algorithm for computing linear approximation for the partitioned circuit blocks: In this attack, the individual partitions from $Part_{all}$ are taken as input from Algorithm 1. For each partition LB_L, a set of outputs of logic gates that sensitizes the key gates in LB_L, is constructed. Subsequently, the set of primary inputs PI_i that sensitizes such outputs of logic gates is computed. These two sets undergone a linear combination of corresponding elements to yield the vector $approx$. For example, consider the partitioned logic block LB_L^1; the set O_{LB_L} for LB_L^1 comprises the output of $\{G16, G10\}$ gates that sensitize ($\rightsquigarrow$) the input gates of LB_L^1. Further, the set I_{LB_L} in step 4 for this logic block comprises the union of:

(i) set of primary inputs $\{I_2, I_3, I_6\}$ which sensitize gate $G16$ and,
(ii) set of primary inputs $\{I_1, I_3\}$ that sensitize gate $G10$.

Algorithm 1 Circuit Partitioning Algorithm: Returns $Part_{all}$ that comprises all partitions LB_L^j; each LB_L^j comprises at least one key gate from a logic cone LC_L^i.

1: **procedure** PARTITION($V, E, KG \in \mathcal{G}_{LC_L^i}$) ▷ Partition LC_L^i into LB_L^j
2: $Part_{all} \leftarrow \phi$ ▷ $\{LB_L^1, LB_L^2, \ldots\}$
3: **for** each O_i, $0 \leq i \leq (m-1)$ **do** ▷ For each PO
4: $j \leftarrow 0$ ▷ j indicates partition number
5: $PS_V^j \leftarrow \phi$ ▷ Set of gates in LB_L^j
6: $PS_K^j \leftarrow \phi$ ▷ Set of key gates in LB_L^j
7: $PS_E^j \leftarrow \phi$ ▷ Set of edges in LB_L^j
8: **while** $(KG \neq \phi)$ **do** ▷ All key gates allocated
9: **while** $(|KG \cap PS_K^j| == 0)$ **do** ▷ No other key gate in LB_L^j
10: **if** $((v \in V) \wedge ((v \in O_i) \vee (v \in KG)))$ **then**
11: $PS_V^j \leftarrow PS_V^j \cup \{v\}$
12: $V \leftarrow V \setminus \{v\}$
13: **for** $((v_j, v) \in E)$ **do**
14: $PS_E^j \leftarrow PS_E^j \cup \{(v_j, v)\}$
15: $E \leftarrow E \setminus \{(v_j, v)\}$
16: **if** $((v \in V) \wedge (v \in KG))$ **then**
17: $PS_K^j \leftarrow PS_K^j \cup \{v\}$
18: $KG \leftarrow KG \setminus \{v\}$
19: $j \leftarrow j + 1$
20: $LB_L^j \leftarrow \{(PS_V^j, PS_K^j, PS_E^j)\}$
21: $Part_{all} \leftarrow Part_{all} \cup LB_L^j$
22: **return** $Part_{all}$ ▷

Both $G16$ and $G10$ drive the input gates $G29$ and $G24$ of LB_L^1, respectively. For this logic block, the linear approximation involves these primary inputs and the output signals of the set O_{LB_L}.

IV. Proposed Differential Attack on LL Circuits

In this attack, we mount a differential attack along with a sensitization attack on a logic locked circuit. As an example of sensitizing the key to the output, consider the circuit in Fig. 4(a); setting $I_3 = 0$ sets the output of the NAND gate to logic 1. From Table I, consider the propagation of differential ΔI to the output ΔY under the effect of key K. As the table demonstrates, the key K affects the polarity of the differential as it propagates from the input of XOR key gate to its output. In Figure 4(a), constraining I_3 propagates the polarity of the differential in $I_1 \wedge I_2$ to the output Y, thereby leaking information of key K to the output Y.

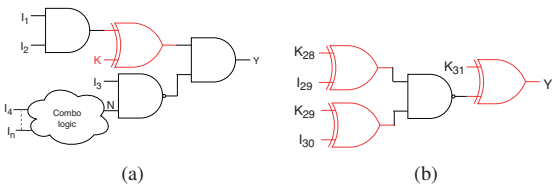
In general, an attacker computes the conjunctive normal form (CNF) representation of the logic locked circuit. For instance, the conjunctive normal form (CNF) representation of the circuit in Figure 4(a) is,

$$(\neg I_1 \vee \neg I_2 \vee \neg K) \wedge (I_1 \vee K) \wedge (I_2 \vee K) \wedge (\neg I_3 \vee \neg N) \quad (1)$$

The attacker first constrains the inputs to the logic values, $I_3 = 0$, $I_1 = 1$, $I_2 = 1$, that sensitizes the key to the output as $Y = \neg K$. For the input set, $I_1 = I_2 = I_3 = 0$, and output

Algorithm 2 Linear Approximation Algorithm for each partition $LB_L \in Part_{all}$

procedure LINAPPROXLB($Part_{all}$, I_i $(1 \leq i \leq n)$, $O_j (1 \leq j \leq m)$)
  for each $LB_L \in Part_{all}$ **do** ▷ For each LB_L in LC_L
    $O_{LB_L} \leftarrow \{O_i\}$ ▷ $O_i \rightsquigarrow K, K \in PS_K^j$
    $I_{LB_L} \leftarrow \{PI_j\}$ ▷ $PI_j \rightsquigarrow O_i, O_i \in O_{LB_L}$
    $approx \leftarrow \bigoplus_{PI_j \in I_{LB_L}} PI_j \oplus \bigoplus_{O_i \in O_{LB_L}} O_i$
    $GS_1 \leftarrow \{I_{LB_L}\}$ ▷ $\forall I_{LB_L}, I_{LB_L} \rightsquigarrow O_{LB_L}$
    $GS_2 \leftarrow \{I_{LB'_L}\}$ ▷ $\forall LB'_L, LB_L \rightsquigarrow LB'_L$
    $ActS \leftarrow \{GS1 \cup GS_2\}$
    $\forall I \in ActS, O'_{LC} \leftarrow LC'_L(I)$ ▷ $LC'_L = LB_L \cup \bigcup_i LB^i_L$, $\forall LB^i_L, LB_L \rightsquigarrow LB^i_L$
    Compute $Pr[O_{KG}|LC_i]$ from O'_{LC}
    for each $K \in PS_K$ **do**
      $Count \leftarrow 0$
      for each $PI_i \in I_{LB_L}$ **do**
        if $approx == 0$ **then**
          $Count \leftarrow Count + 1$
      $Count[K] \leftarrow Count$
    $CorrectKey \leftarrow K' : |K' = \max_{K \in PS_K} \{\frac{Count[K]}{2^n} - \frac{1}{2}|\}$ with $Pr[O_{K'}|LC_i]$



(a) (b)

Figure 4. a) Logic locked circuit with a key gate and some inputs close to the output. b) Another example of logic locked circuit.

Table I
POLARITY INVERSION (RESTORE) OF AN OUTPUT DIFFERENTIAL W.R.T AN INPUT DIFFERENTIAL AT A KEY GATE OCCURS IF THE ACTUAL KEY VALUE IS 1(0).

K	ΔI	ΔY
0	$0 \rightarrow 1$	$0 \rightarrow 1$
0	$1 \rightarrow 0$	$1 \rightarrow 0$
1	$0 \rightarrow 1$	$1 \rightarrow 0$
1	$1 \rightarrow 0$	$0 \rightarrow 1$

$Y = 0$, we obtain $K = 0$ from the CNF expression. For circuits, wherein sensitizing does not work depending on the circuit topology, we apply differential attack.

In differential attack on logic locking techniques, we create differentials at the input of the logic circuit that propagate to the output to recover the key. In this attack, we constrain certain inputs to constant values while other inputs are subjected to input differentials ΔI whose propagation to a logic cone output is monitored. An *input differential*, ΔI of input I is defined as $\Delta I = I_1 \oplus I_2$, i.e., the XOR of two consecutive input values that I is subjected to. The differential attack is based on the property of a XOR key gate that the value of the

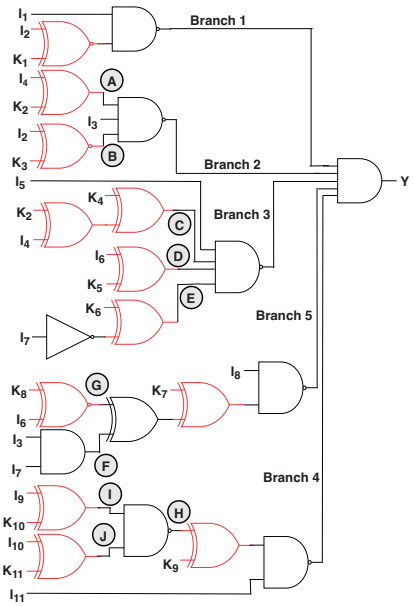


Figure 5. A circuit example for differential attack. The key gates are shown in red color and the node labels are encircled.

secret key affects the propagation of differential polarity from the input to the output. This phenomenon is exhibited by the following property of a XOR key gate whose inputs are I and K while output is Y.

We consider the logic locked circuit shown in Figure 5. In this circuit, we choose the smallest logic cone, i.e., the logic cone that comprises minimum number of key gates. The entire circuit is partitioned until all gates of the circuit have been identified. For any logic locked circuit under attack, we partition the circuit into branches and subbranches. In Figure 5, the five branches of the circuit have been shown. We constrain the maximum number of branches to logic 0 or logic 1 depending on the circuit topology. Subsequently, we apply three attack modes depending upon whether an attack mode recovers the key in the partition. we present all the three attack modes with respect to circuit in Figure 5.

A. Attack Mode 1

We apply constant inputs and attempt to determine the key input values from the input values, output values, and the CNF expression of the partitioned circuit. For instance, in circuit shown in Figure 5, we constrain $Branch2$, $Branch3$, $Branch4$, and $Branch5$ to logic 1 by subjecting the inputs, $I_3 = I_5 = I_8 = I_1 = 0$, respectively. Further, we apply the inputs $I_1 = I_2 = 1$. his reduces the CNF expression to $\neg K_1 = 0$, i.e., we recover $K_1 = 1$.

B. Attack Mode 2

We create an input differential ΔI to a partitioned circuit and observe the polarity of the output differential ΔY to obtain the key values. We compute a subset of primary inputs $(I' \subset I)$, such that a non-zero differential $\Delta I'$ creates a non-zero output differential ΔY in the partitioned circuit. The

propagation of ΔY is traced to the output of the logic cone. In Figure 5, we constrain $Branch1$, $Branch3$, $Branch4$, and $Branch5$ to logic 1 through input pattern, $I_1 = I_5 = I_8 = I_{11} = 0$, respectively. We target $Branch2$ and set $I_3 = 1$. In this case, if $I_2 == K_3$, then $I_4 \rightsquigarrow Y$, else $I_4 \not\rightsquigarrow Y$. If $I_2 = K_3$, the node B in $Branch2$ is set to logic 1, else it is set to logic 0. So the pair of signals (I_2, I_4) determine the value of K_3 in this case. If gate driven by nodes A and B occur to be logic gate X, then the logic value at node B can be determined as,
$B = \Delta I \rightsquigarrow \Delta Y?P : Q$, (X gate must be in the active branch). $If\ X = NAND$ or AND, then $P = 1, Q = 0$. If $X = OR$, then $P = 0, Q = 1$. The logic value at node B yields the relation between I_2 and K_3. Subsequently, we get the value of K_2 from Attack Mode 1 by applying input constraints on $Branch2$.

C. Attack Mode 3

In case both $Attack\ Mode\ 1$ and $Attack\ Mode\ 2$ fail, we vary all the inputs to an active sub-branch to yield relations between the key inputs and intermediate wire values. Subsequently, we again apply $Attack\ Mode\ 1$ and $Attack\ Mode\ 2$ on each of the active sub-branches that yields the relations in the previous step. The results of the mounted differential attack in presented in the next section.

V. Experiments and Results

A. Linear Approximation Attack

For the two partitions, LB_L^1, and LB_L^2 shown in Figure 3, we obtain the following linear approximation equations that yields minimum Hamming distance with the corresponding functions of the respective blocks,

$$G10 \oplus G16 \oplus I_1 \oplus I_3 \oplus I_6 \oplus I_2 \oplus K_5 \oplus K_2 \oplus K_3 = 0$$
$$G11 \oplus I_3 \oplus I_6 \oplus K_2 = 0$$

In these equations, $G10$, $G16$, and $G11$ represent the output of the respective gates in the circuit. The partition LB_L^1 comprises two keys K_1 and K_6, which have four possible values. For this circuit there are 32 possible values of the input. The linear bias for the joint probability of occurrence of possible key values for (K_1, K_6) are shown in Table II. From the table, the linear bias profiles of the two key inputs reduce the key search space by half.

Table II
Linear bias for occurrence of key values for partition LB_L^1

K_1	K_6	$Pr[\frac{Count}{2^n}]$	$Bias = Pr[\frac{Count}{2^n}] - \frac{1}{2}$
0	0	$\frac{18}{32}$	$\frac{1}{16}$
0	1	$\frac{14}{32}$	$-\frac{1}{16}$
1	0	$\frac{14}{32}$	$-\frac{1}{16}$
1	1	$\frac{18}{32}$	$\frac{1}{16}$

The partition LB_L^2 comprises the key K_5, which has two possible values 0 and 1, respectively. The linear bias for the for the probability of occurrence of K_5 is as shown in Table III. The linear bias profile shows that the two key values are not equiprobable. This sets the criteria that the key gate locations should be determined in such a way that linear bias profiles should be equiprobable for all possible values of the key.

Table III
Linear bias for occurrence of key values for partition LB_L^2

K_5	$Pr[\frac{Count}{2^n}]$	$Bias = Pr[\frac{Count}{2^n}] - \frac{1}{2}$
0	$\frac{14}{32}$	$-\frac{1}{16}$
1	$\frac{18}{32}$	$\frac{1}{16}$

The linear approximation expression for the partition LB_L^3 does not comprise any key input. Hence, from Table IV, we find that the linear bias profiles for the key inputs K_2 and K_3 in the last partition LB_L^3 are equiprobable. Hence, the key gate locations in this partition is resilient against linear approximation attack.

Table IV
Linear bias for occurrence of key values for partition LB_L^3

K_2	K_3	$Pr[\frac{Count}{2^n}]$	$Bias = Pr[\frac{Count}{2^n}] - \frac{1}{2}$
0	0	$\frac{16}{32}$	0
0	1	$\frac{16}{32}$	0
1	0	$\frac{16}{32}$	0
1	1	$\frac{16}{32}$	0

B. Differential Attack

In the circuit shown in Figure 5, we set $Branch1$, $Branch2$, $Branch4$, and $Branch5$ to logic 1 by putting $I_1 = I_2 = I_8 = I_{11} = 0$, respectively. We set $I_5 = 1$; if $Y = 0$, then logic value at nodes $C = D = E = 1$. This implies that the value of $Branch3$ to zero. Applying $Attack\ Mode\ 1$, we get $K_4 = 0$ (from C), $K_5 = 0$ (from D), $K_6 = 0$ (from E).

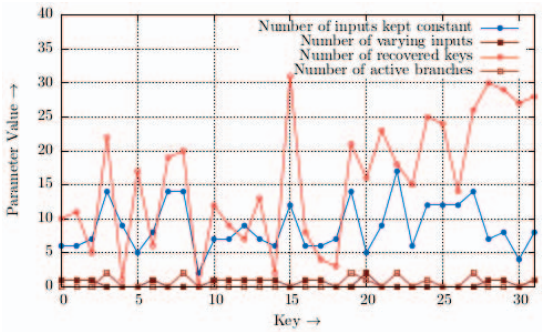
Further, in the circuit we constrain $Branch1$, $Branch2$, $Branch3$, $Branch5$ at logic value 1 by setting $I_1 = I_3 = I_8 = I_5 = 0$, respectively. We set $I_{11} = 1$. For $Y = 0$, one of the following input sets need to be asserted, $I9 = I10 = 0, I11 = 1$, $I_9 = 0, I_{10} = I_{11} = 1$, or $I_{10} = 0, I_9 = I_{11} = 1$. If we create a non-zero differentials ΔI_9 and ΔI_{10}, then they also sensitize the output differential too, i.e., $\Delta I_9 \rightsquigarrow Y$. This yields that nodes $I = J = 1$, hence $K_{10} = 1$ and $K_{11} = 0$. In the subsequent step, we recover K_9 by applying $Attack\ Method\ 1$.

We mounted this attack on logic locked versions of $c432$ ISCAS'85 benchmark circuit that employs a 32-bit key input. The attack results demonstrated in Figure 6 indicates that the differential attack:

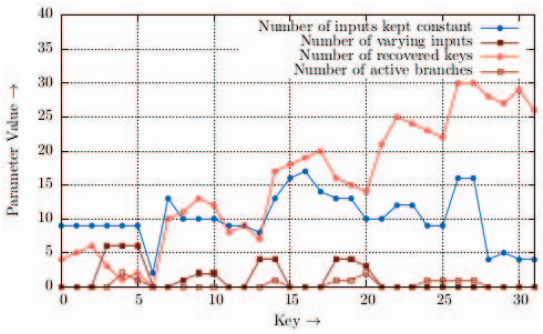
(i) on RLL technique recovers lesser numbers of key values in each iteration as shown in Figure 6(a). Also, the number of inputs that need to be set in order to mount a successful key recovery attack is higher than 15 for correct key input value $K = 22$.

(ii) on FLL technique recovers larger number of keys per iteration. From Figure 6(b), the red circled plot demonstrates that each attack mode iteration yields three key input values.

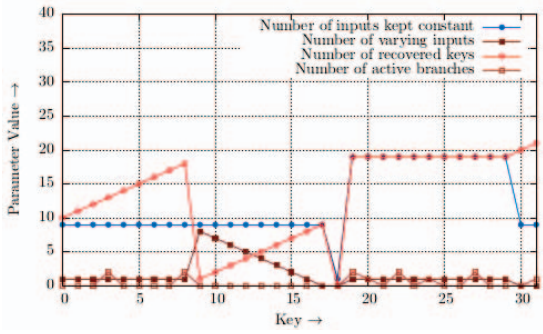
(iii) on SLL technique recovers relations between the key inputs. As shown in Figure 6(c), this attack requires increased number of iterations to recover a key value. Also mounting differential attack requires setting a large number of primary inputs to a constant value.



(a)



(b)



(c)

Figure 6. Differential attack results on (a) random logic locked (RLL), (b) fault-analysis based logic locked (FLL), and (c) strong logic locked (SLL) versions on c432 ISCAS'85 benchmark circuit. The results indicate the number of inputs that need to be kept constant and required to vary or produce a differential to yield each key values. Further, it depicts the number of previously recovered key values that were used to extract a key.

VI. Conclusion

In this paper, we demonstrate that the key gate locations in a logic locked circuit may render a certain partition of the circuit vulnerable to linear approximation attack. The linear bias in the probability of occurrence of certain key values leaks information to the attacker that all key values are not equiprobable, hence informs the attacker of the hot-spots in a logic locked circuit. In differential attack, we demonstrate how the propagation of input differential across a XOR key gate is affected by the value of key input. If the propagation of this differential from the primary inputs to the output of a logic cone is observable to an attacker, then the key information gets leaked to him.

References

[1] M. Yasin and O. Sinanoglu, "Evolution of logic locking," in *Very Large Scale Integration (VLSI-SoC), 2017 IFIP/IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[2] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of Untegrated Circuits," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 1069–1074.

[3] U. Guin, D. Forte, and M. Tehranipoor, "Anti-counterfeit techniques: From design to resign," in *14th International Workshop on Microprocessor Test and Verification, MTV 2013, Austin, TX, USA, December 11-13, 2013*, 2013, pp. 89–94.

[4] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, 2016.

[5] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012*, 2012, pp. 953–958.

[6] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Trans. Computers*, vol. 64, no. 2, pp. 410–424, 2015. [Online]. Available: https://doi.org/10.1109/TC.2013.193

[7] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A Novel Hardware Logic Encryption Technique for Thwarting Illegal Overproduction and Hardware Trojans," in *On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International*. IEEE, 2014, pp. 49–54.

[8] Y. Xie and A. Srivastava, "Mitigating SAT Attack on Logic Locking," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 127–146.

[9] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," *DAC Design Automation Conference 2012*, pp. 83–89, 2012.

[10] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 137–143.

[11] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," in *Hardware Oriented Security and Trust (HOST), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 95–100.

[12] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security Analysis of Anti-SAT," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 342–347.

[13] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "Ttlock: Tenacious and traceless logic locking," in *Hardware Oriented Security and Trust (HOST), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 166–166.

[14] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1601–1618.