A Novel PUF based Logic Encryption Technique to Prevent SAT Attacks and Trojan Insertion

Soraya Mobaraki
Department of Computer Engineering
University of Guilan
Rasht, Iran
s_mobaraki@webmail.guilan.ac.ir

Amirata Amirkhani
Department of Computer Engineering
University of Guilan
Rasht, Iran
amirata@msc.guilan.ac.ir

Reza Ebrahimi Atani
Department of Computer Engineering
University of Guilan
Rasht, Iran
rebrahimi@guilan.ac

***Abstract*—The manufacturing of integrated circuits (IC) outside of the design houses makes it possible for the adversary to easily perform a reverse engineering attack against intellectual property (IP)/IC. The aim of this attack can be the IP piracy, overproduction, counterfeiting or inserting hardware Trojan (HT) throughout the supply chain of the IC. Preventing hardware Trojan insertion is a significant issue in the context of hardware security (HS) and has not been considered in most of the previous logic encryption methods. To eliminate this problem, in this paper an Anti-Trojan insertion algorithm is presented. The idea is based on the fact that reducing the signals with low-observability (LO) and low-controllability (LC) can prevent HT insertion significantly. The security of logic encryption methods depends on the algorithm and the encryption key. However, the security of these methods has been compromised by SAT attacks over recent years. SAT attacks, can decode the correct key from most logic encryption techniques. In this article, by using the PUF-based encryption, the applied key in the encryption is randomized and SAT attack cannot be performed. Based on the output of PUF, a unique encryption has been made for each chip that preventing from counterfeiting and IP piracy.**

Keywords—SAT Attack, Logic Locking, Rare Signal, Hardware Trojan, Hardware Obfuscation, Design-For-Trust

I. INTRODUCTION

Nowadays, most of integrated circuit design companies do not have manufacturing foundries and outsource manufacturing and production of IC in order to reduce costs. This opens the way for attackers that can make reverse engineering attacks, IC/IP piracy, overproduction, and counterfeiting. Therefore, by reverse engineering and knowing the function of the circuit, they can insert a malicious circuit which is called Hardware Trojan (HT). The HT can be used to gain information leakage and manipulation of circuit function [1]. To prevent these attacks different techniques have been presented, including logic based hardware obfuscation. In logic based hardware obfuscation, the techniques such as obfuscation by using logic encryption and camouflaging have been introduced to prevent IP piracy, counterfeiting, overproduction and hardware Trojan insertion [2]. The logic encryption hides the correct function of the IC by using the proper key in the design. In other words, the circuit has a correct function only when a correct key is used. There are typically two kinds of encryption: first, a new gate is inserted in the design and second, a gate is replaced with another gate. The overhead of the first method is higher compared to the second one from power, delay, and area points of view. In the insertion method, the gates can randomly insert in the netlist [2], which cause hardware overhead. Thus, the choice may be based on a 50% hamming distance between the output generated by the correct key and wrong key [4]. Encryption can be done by inserting the AND/OR logic gate [5]. The second method, if the replacement gate is optimized, has less overhead than the first method. A new logic encryption is the replacement of the gate with LUT [6]. In [7], a new topology introduced for logic gates that have the area, power and delay overhead less than previous methods. This new topology is replaced with selected gates for logic encryption. They also considered the controllability of signals to prevent HT insertion and easily detect them. Therefor, the reduction of signal with the LC has more efficiency to prevent HT insertion. However, the problem of signals with LO has remained that the attacker can easily insert HT in them. Another important issue in encryption methods is preventing SAT attacks. The attacker can identify the encryption key by using the relationship between the output and the encryption key; without applying a brute force attacks which has an exponential time complexity [8].

In this paper we improved the REAL algorithm presented in [7] and in addition to controllability, the observability of the signals is laso taken into account for gate replacement candidates. Besides PUF based key generation strategy is applied for logic encryption which is random and can be used as a prevention technique against SAT attacks. For these attacks, the attacker must have a copy of the netlist of obfuscated circuit. An attacker may be at the foundry and have access directly to the netlist or after buying a chip by using the reverse engineering techniques, the netlist is obtained. For a SAT attack, the keys must be available as inputs, as well as obfuscation gates represented as a key-programmable gate (KPG). If the inputs are represented by X and the outputs by Y, the C circuit will be $C(X, Y)$ and $C(X, K, Y)$ is a circuit that is obfuscated by the key K. A correct key leads us to $C(X, Y) = C(X, K, Y)$. Two keys K_1 and K_2 are considered for the attack. The output of both keys is compared for each input value X. If and only if one of the inputs is found that their output is not equal, the comparison ends and this input called a distinguishing input (DIP). In this case, for inputs $X_i = (x_0, x_1, \ldots, x_{x-1})$ the output of the IC is compared with the outputs of K_1 and K_2, either one or both keys may be wrong. Therefore, the wrong key will be remove from the correct keys set. Each new key is evaluating with the DIPs; K_1 and K_2 are equivalent when the outputs of them are the same for all inputs. The key is correct when the DIP is not found and its output is correct for all inputs [8].

Due to the process variation in chips, physical parameters are different even in the same chips with the same technology and fabrication. These intrinsic differences are used in the conceptual term called "Physical Unclonable Function" (PUF),

978-1-5386-8274-6/18/$31.00 ©2018 IEEE

which generate the random numbers and is used for authentication and encryption. These random numbers are sometimes used for keys in encryption algorithms. In logic encryption, security of the key is very important. An attack to this key by an adversary will lead to the IP of all the IC. Therefore, in this paper, a unique intrinsic PUF based key for logic encryption of each chip is applied.

The rest of the paper is organized as follows: Section II provides related works to the logic encryption, and the use of SAT attacks for decryption. Section III presents observability and controllability tests for preventing HT insertion. The PUF properties used to produce the encryption key and prevention techniques for IP/IC piracy and counterfeiting of chips is also explained in this section. Section IV describes the simulation and evaluation results and finally, section V concludes the paper.

II. Related Works

In logic encryption, gate insertion such as *XOR/XNOR* can be used to select which one of the inputs can be considered as the key and according to the value of the key, output signal will be buffered or reversed. Random insertion method in [2] encrypt faster than other methods but it is a weak encryption method because for some inputs, the output will be generated correctly for the wrong key. In [4], encryption is performed based on the achievement of a 50% hamming distance between the output generated by the correct key and the output generated by the incorrect key. Another insertion method, *mux2×1* is used for encryption that the key is used as the mux selector. In mux, one of the inputs is used as the correct signal, which is selected based on the highest fault impact, and another input used as the wrong signal, which is selected based on the contradiction metric. This metric is used for showing the maximized difference between the correct value and the incorrect value. But this metric cannot show that the correct and incorrect amounts are fully complementary [9]. There are reports about vulnerability of all of these techniques against the SAT attack [10], [7]. It is important to mention signals with low-controllability and low-observability cannot prevent the insertion of Trojan and this fact has not been considered in [2], [4]. The idea of *AND/OR* gate insertion was presented in [5] to reduce signals with low-controllability, although it cannot completely eliminate low-controllability signals. An example is shown in Fig. 1, where computes the probability of logic '0' (P_0) and '1' (P_1) on each signal of outputs of the gates by considering P_0/P_1 , for example, G_3 and G_4 gate outputs are rare.

In Fig. 2, by inserting the OR gate in the output of G3, the probability of "1" and "0" output is close together but the input of G3 still has low controllability.

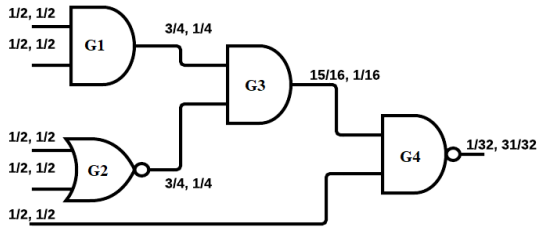


Fig. 1. Original Circuit

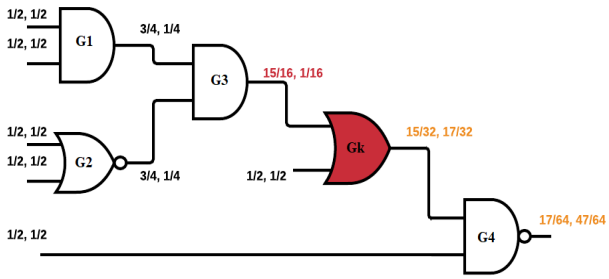


Fig. 2. Encrypted circuit using inserted OR gate.

In methods that use the gate replacement instead of insertion of a new gate, encryption key is hided in the structure. This structure can be a LUT that hides the function and creates a lot of confidentiality and makes it hard to find the correct key, but it has a lot of silicon overhead [6]. The LUT can be based on reconfiguration, that with the increase in the number of keys, increases area overhead [11]. It is important to mention that that necessarily by increasing the number of keys, it does not increase security against SAT attacks. Attempt to select the appropriate location for LUT and consider the critical path cause reduction in the delay and area overhead. By reducing the signals with low controllability (decreased the number of nodes with high Skew Probability Signal (SPS)) helps to prevent HT insertion [12]. However, one should not forget that replacement the gate with LUT, creates a large area overhead and selection the LUT to replace have a large area, power, delay overhead and will not be an effective replacement. In [7], a gate structure (key-gat topology) with a much less area than the LUT was proposed which reduces the number of low-controllability signals. Their idea is reducing low-controllability signals and try to equalize the probability of "0" (P_0) and the probability of "1" (P_1) in the output of the gate. For example, the output of *AND\NOR* gate that has the P_0 less than P_1, they increase the value of P_1 for the incorrect key. In the presented key-gate topology, the *AND/NOR* gate has a valid key (K=1) and for the (K=0), generates constant "1" in output. The *OR/NAND* gate, which has a valid key (K=0) and P_0 is less than P_1, provides a constant value "0" for the incorrect key (K=1). The *XNOR/XOR* gate, for each probability in the input gate, always produces the equal P_0 and P_1 in outputs. The *NOT* gate inverts the input so the value of P_0 in the output is equal to P_1 in the input and vice versa, the value of P_1 in the output is equal to P_0 in the input. The *NOT0/NOT1* in new topology, for each probability of the input, produces P_0 close to P_1 in the output. Thus, *NOT0/NOT1* based on the fact that P_0 is smaller in the output or P_1 is selected. *NOT1* has the correct key (K=1) and is selected when the value of P_1 is less than P_0 in the output. And *NOT0* has a valid key (K=0) and it is selected when the value of P_0 is less than P_1 in the output. The *AND/NOR* key-gate topology and its transistor level structureis shown in 0.

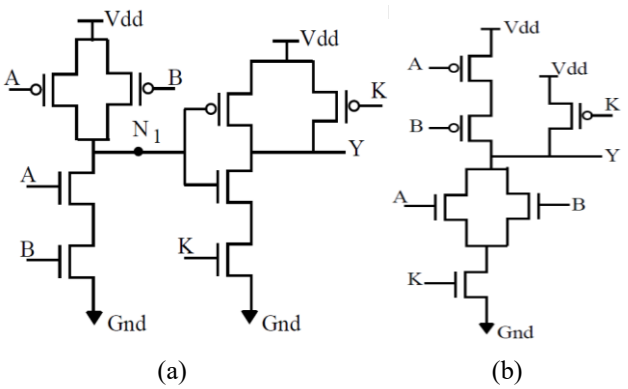


Fig. 3. (a) AND gate (b) NOR gate [7]

The correct key for each key-gate is shown in TABLE Iand TABLE II

TABLE I. THE GATES OUTPUT PRESENTED IN [7]

Key	Inputs		Output				
k	*A*	*B*	*AND*	*NAND*	*OR*	*NOR*	*XOR/XNOR*
0	0	0	1	1	0	1	0
0	0	1	1	1	1	1	1
0	1	0	1	1	1	1	1
0	1	1	1	0	1	1	0
1	0	0	0	0	0	1	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	1

TABLE II. INVERTER GATE OUTPUT IN TOPOLOGY [7]

Key	Inputs	Output	
k	*A*	NOT_1	NOT_0
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

The REAL algorithm presented in [7] calculates the value of the controllability for all the signals, which is shown by the vulnerable factor VF=P_0-P_1. The gates that have VF greater than threshold values are considered as vulnerable gates. The initial replacement is done for all the gates; the gate that reduces more vulnerable gates is selected as a critical gate. Nevertheless, the problems have remained 1) by reducing the number of signals with low-controlling (high VF), the problem of the signals with low-observability is still remained. Because the attacker does not just focus on rare signals to insert a Trojan, it may choose a signal that has low-observability. 2) The presented key expansion idea which is used to reduce the number of keys is ineffective. Because the used key in the topology has $P_0=P_1=1/2$ and, therefore, the output of gate must be set to $P_0=P_1=1/2$, that can be used as a key for the next key-gate without increasing VF. 3) It is not clear how to choose the candidate between gates reducing the equal number of vulnerable gates. 4) Most encryption methods are weak against SAT attacks. So far many solutions were presented preventing SAT attacks, for example in [10], in the interference graph, clique with the maximum size of non-mutable keys created. Therefore, the attacker is inevitable to use brute force attacks. However, many of Anti-Sat Attack methods, the Skew Probability Signal (SPS) or controllability of signals has not been considered and known vulnerable in [13].

III. PROPOSED ALGORITHM

In this paper, in addition to considering the reduction of signals with low-controllability, the observability of the signals is also taken into account and calculated and tried to reduce the number of low-observability signals. With respect to the amount of observability and controllability of the signals, a weight function (WF) is obtained and according to this WF, encryption is performed by the replacement method. In this paper SAT attacks were also eliminated using PUF based encryption keys. Finally, in order to prevent the counterfeiting and IP piracy of the chips, a unique encryption pattern will be provided based on the output of the PUF for each chip.

A. Increasing Observability to Prevent Trojan Insertion

Although logic encryption has several advantages its application alone might not properly minimize rare signals and will not be a good technique to prevent HT insertion. This is due the cases that attacker may insert the trojan in a signal with low-observability feature and its effect is less on the output. In this paper, a new algorithm in logic encryption is presented, which has Anti-Trojan insertion property. The presented topology in [7] is used as a replacement gate. 0-Controllability in a signal is equal to P_0 and the 1-controllability is equal to P_1. According to equation (1), the overall controllability of a signal is obtained from the multiplication of the 0-controllability (C_0) and the 1-controllability (C_1) in the signal:

$$CS=C_0\times C_1 \quad (1)$$

In this paper by using the presented STAFAN method in [14], controllability in the output of the gate is obtained from C_1 and C_0 in the inputs of the gate. On the other hand the observability is obtained from the output to the inputs and observability of the main outputs of the circuit is "1". The observability in one input of the gate can be achieved by observability of the output and controllability of other inputs. For example, consider the AND gate in Fig. 4.

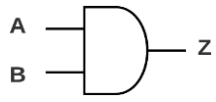


Fig. 4. AND gate

According to equation (2), the C_1 in the output of AND is obtained when both inputs are "1".

$$C_1(z)=C_1(a).C_1(b) \quad (2)$$

And in the equation (3) to calculate C0, it just need to calculate the complement of C0. In the equation (4), observability of the inputs "a" is obtained. In the input "b", observability is calculated by the equation (5).

$$C_0(z)=1-C_1(z) \quad (3)$$

$$O(a)=O(z).C_1(b) \quad (4)$$

$$O(b)=O(z).C_1(a) \quad (5)$$

Another example is *OR* gate in which the observability and controllability can be calculated such as the *AND* gate. To calculating the C_0 in the output of *OR* gate, used the equation (6). C_1 is also calculated by the equation (7).

$$C_0(z)=C_0(a).C_0(b) \quad (6)$$

$$C_1(z)=1-C_0(z) \quad (7)$$

The observability in the input "a" of the *OR* gate is calculated by equation (8). Observability in the input "b" is also calculated by equation (9).

$$O(a)=O(z).C_0(b) \quad (8)$$

$$O(b)=O(Z).C_0(a) \quad (9)$$

The observability of other gates is similar the *AND* and *OR* gate and used the observability of output to calculate the observability of the inputs. Controllability is calculated from the input to the output. In this paper, a new replacement method is proposed that removes vulnerable signals in addition to encryption. At the beginning of the Algorithm I, the number of key-gate for replacement in each circuit is determined. The replacement continues until the number of replacements (NOR) in the last round becomes zero. The first step is to input the circuit netlist (CN) to the algorithm. Then controllability (SC) and observability (SO) of the signals is calculated. A signal with lower SC compared to the threshold of controllability (SC_{th}) is considered as a signal with the first vulnerability factor (FVF), and the number of signals with the first vulnerability factor (NFVF) factor is determined. The signal with lower SO compared to the threshold of observability is considered as signal with the second vulnerability factor (SVF). After that we determine the number of signals with the second vulnerability factor (NSVF). In the first replacement, the NFVF and NSVF values are considered as the number of signals with the first vulnerability factor (ONFVF) and the number of signals with second vulnerability factor (ONSVF) in the original circuit. Afterward, an initial replacement is done for each gate and the values (NFVFt) and (NSVFt) are considered as the number of signals with the first vulnerability factor and the number of signals with the second vulnerability factor per initial replacement, respectively. The difference between the number of vulnerability signals after the initial replacement and the original circuit is calculated and as (diff_FVF) and (diff_SVF) are considered. Based on the weight function (WF) of the equation (10), a comparison is done between replacements.

$$WF= diff\_FVF + diff\_SVF \quad (10)$$

If *WF* is equal to zero, this means that initial replacement cannot improve any vulnerability factors, if it is negative, it destroys one or both factors. To avoid these alternatives, only the gates in the (Gate_list) that have improved at least one of FVF or SVF is specified and a list of weights (WF_list) each initial replacement that improved vulnerability factors is determined. Thus, lists (SCF_list) and (SOF_list) considered, which indicate the improvement in controllability and observability of signals in each replacement, respectively. For each replacement licensed in NOR, the value (diff_SCF = ONFVF – NFVF*)* and (diff_SOF = ONSVF – NSVF*)* is calculated. Based on the fact that in the previous replacement, which one of the vulnerability factors has more improvment, in this replacement, it's trying to improve another factor. If it was the first replacement, diff_SOF and diff_SCF are both zero, because both NSVF and NFVF are equal to ONFVF and ONSVF in the original circuit. If two factors are improved equally (diff OF = diff SCF), selected the gate from the Gate_list that has the maximum weight in WF_list. And if there are several gates with maximum weights, by reviewing SCF_list and SOF_list, among them, the gate is selected that fully or approximately improved both factors. And the gate is inserted in a set of selected gates (SG) for replacement. And if the FVF more improved (diff_SCF> diff_SOF), in this replacement SVF should more improve. In this case, the maximum value in the SOF_list is selected and if several gates had the maximum value, the gate with the maximum weight in WF_list has selected and inserted in the SG. Finally, if the SVF has been improved in the previous replacement, the FVF should be improved in this replacement. For example, assume that the following lists are available:

- Gate_list = [1,2,3,4,5,6]
- SCF_list = [2,3,3,1,0,0]
- SOF_list = [2,1,0,3,4,1]
- WF_list = [4,4,3,4,4,1]

In this example, the vulnerability factors have similar improvement in the previous replacement and in this replacement, four gates have the weight equal to 4, that maximum weight. One of these four gates according to the SCF_list and the SOF_list is selected. The first gate which have improved both the factors equally is selected. The algorithm terminates when 1) the desired number of replacement is done 2) no longer has a better answer. In this case, the algorithm returns the obfuscated netlist. The execution time to calculate the number of each vulnerable factor is *O(N)*. The replacement of each gate is also performed in *O(1)* and for all circuit gates, an initial replacement is performed with *O(N)*. Finally, the runtime complexity of the algorithm is *O(N2)= N.(1+N+N)*.

B. Using PUF to Produce the Encryption Key

Using the PUF for the encryption key makes the SAT attack more difficult. In this paper, for each key-gate, one bit of the output of PUF is used as the key for the key-gate. In the Anti-Trojan insertion algorithm, after identifying the selected gate for replacement, the key-gate is determined based on the gate library which presented in [7]. The correct key for each key-gate is shown in TABLE Iand TABLE II.

Algorithm I: proposed Anti-Trojan Insertion Algorithm

```
1:  Input: SCth, SOth, NOR, CN
2:  Output: ON
3:  Function: ATIA(CN, SCth, SOth, NOR)
4:      ON ← CN
5:      FOR i = 1 to NOR DO
6:          NFVF ← get_num_FVF(ON, SCth)
7:          NSVF ← get_num_SVF(ON, SOth)
8:          IF i = 1 THEN
9:              ONFVF ← NFVF
10:             ONSVF ← NSVF
11:         END IF
12:         FOR EACH gate in ON DO
13:             ONt ← replace (ON, node)
14:             NFVFt ← get_num_FVF(ONt, SCth)
15:             NSVFt ← get_num_SVF(ONt, SOth)
16:             diff_FVF ← NFVF - NFVFt
17:             diff_SVF ← NFVF - NFVFt
18:             WF ← diff_FVF + diff_SVF
19:             IF WF > 0 THEN
20:                 IF diff_FVF >= 0 and diff_SVF >= 0 THEN
21:                     Gate_list ← Gate
22:                     SCF_list ← diff_FVF
23:                     SOF_list ← diff_SVF
24:                     WF_list ← WF
25:                 END IF
26:             END IF
27:         END FOR
28:         IF WF_list = null THEN
29:             break
30:         END IF
31:         diff_SCF ← ONFVF - NFVF
32:         diff_SOF ← ONSVF - NSVF
33:         IF diff_SCF = diff_SOF THEN
34:             SG ← get_gate(WF_list, SOF_list, SCF_list,
        Gate_list)
35:         ELSE IF SCF_diff > SOF_diff THEN
36:             SG ← get_gate(WF_list, SOF_list, Gate_list)
37:         ELSE
38:             SG ← get_gate(WF_list, SCF_list, Gate_list)
39:         END IF
40:         ON ← replace (ON, SG)
41:     END FOR
42:     RETURN ON
43: END
```

Each bit of the output of PUF is used as the key for the selected gates for replacement, respectively. Since the output of PUF is randomize, the key generated by PUF for the selected gate that going to be replaced might be different from its real key so the gate is replaced with its equivalent gate in the library. According to TABLE I the *AND* gate has the correct key "1". For example, if the *AND* gate is selected for the replacement, but the key generated by PUF for key-gate is "0". Therefore, instead of the *AND* gate, the *NAND + INV* gates is replaced. The correct key for the *NAND* gate in TABLE I is "0". In other words, in each chip based on bits generated by the PUF, there is a unique replacement for the Selected Gates (SG) that is supposed to be replaced. By doing this, the SAT attack becomes difficult to get the correct key because PUF is used to generate the encryption key and for each chip, there is a unique encryption. Therefore, the time to find the relationship between the output of chips and the encryption key will be exponential. The anti-trojan insertion algorithm introduced in the previous section, has been improved to prevent SAT Attack and is presented in Algorithm II.

Algorithm II: Logic encryption based PUF

```
1:  Input: ON, PUF_Key_list
2:  Output: FON
3:  Function: LPFU(ON, PUF_Key_list)
4:      i = 0
5:      FOR EACH gate in ON DO
6:          IF gate = key_gate THEN
7:              IF PUF_Key_list[i] ≠ valid key for gate THEN
8:                  FON ← replace (gate, equal_gate)
9:                  i = i + 1
10:             END IF
11:         END IF
12:     END FOR
13:     RETURN FON
14: END
```

In Algorithm I, the obfuscated netlist will be obtained, which is used in Algorithm II. In line 7, if the key generated by PUF (PUF_Key) is not equal to the correct key of key-gate (obfuscation gate), it is replaced with the equivalent gate. The equivalent gate has an equal key to PUF_Key. In TABLE III based on PUF_Key, its equivalent gates are selected for replacement.

TABLE III. EQUIVALENT GATES TO REPLACE

Gate	Valid key	PUF_Key	Equal gate for replace
AND	1	0	$NAND_{topology}$ + INV
OR	0	1	$NOR_{topology}$ + INV
NOR	1	0	$OR_{topology}$ + INV
NAND	0	1	$AND_{topology}$ + INV
XOR	0	1	$XNOR_{topology}$ + INV
XNOR	1	0	$XOR_{topology}$ + INV

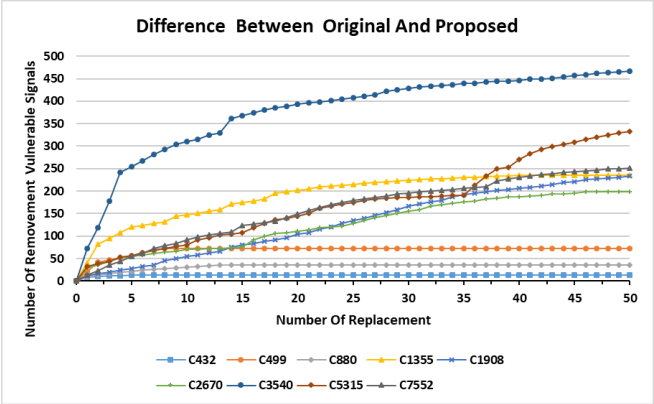
C. Preventing of IP/IC Piracy and Counterfeiting

Algorithm II makes it possible for each chip, has a unique encryption pattern based on output generated by the PUF. For example, if N gate are selected for encryption in a design, this means that N key-gate must be replaced. For each key-gate, it is necessarily to use the one bit of the output of PUF as key. Finally, there are 2N different replacement and different encryption, because the output generated by the PUF will be randomized on each chip. This can prevent from IP piracy, reverse engineering and counterfeiting of chips.

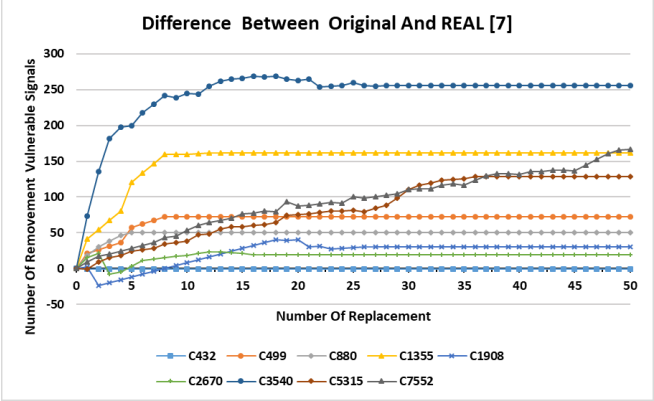
IV. SIMULATION RESULTS AND SECURITY ANALYSIS OF PROPOSED ALGORITHMS

The methods proposed in this paper have been evaluated on the ISCAS-85 benchmarks. Algorithm I will reduce the risk of trojan insertion. In Algorithm II, the replacement of key-gate with equivalent gates does not change the controllability and observability of signals, and the obtained results from Algorithm I to prevent the insertion of the HT. The observability threshold of this evaluation is SOth = 0.07, and

controllability threshold SCth = 0.95×0.05 = 0.0475 the reduction in the number of vulnerable signals in proposed method and presented method in [7] is shown in **Error! Reference source not found.**.



(a)



(b)

Fig. 5. Improvement of vulnerability factors in (a) proposed method and (b) [7]

In **Error! Reference source not found.**(a), by increasing the number of keys, the proposed algorithm reduces the vulnerable signal and in **Error! Reference source not found.**(b), most of the benchmarks, by increasing the number of keys, not improve vulnerable signals. The number of signals with low observability (NOSVF) and low controllability (NOFVF) after replacement with the number of keys presented in [7], as shown in TABLE IV. The number of signals with LO in the proposed method is less than [7]. In **Error! Reference source not found.**, by increasing the number of keys, the number of signals with LC and LO in the proposed method is improved.

In **Error! Reference source not found.**, the number of vulnerable signals is shown in each benchmark. In **Error! Reference source not found.**, the comparison of the average number of vulnerable signals with SO_{th}= 0.0475 and SC_{th}= 0.07 is shown for all benchmarks.

In **Error! Reference source not found.**the result of this work is compared to [7] with different SOth and SCth=0.0475.

The run-time after Algorithm II is $O(N^2)$ because, by merging the two algorithms, after the SG is specified, it can base on the key that generated by PUF, be replaced with equivalent gates. The PUF for generating the encryption key must be carefully selected and have a stable output. There are several choices for the PUF but since this an intrinsic PUF a stable Ring osilator PUF [15] can be applied to generate the keystreams for logic encryption.

TABLE IV. LC AND LO WITH KEY LENGTH PROPOSED IN [7]

Circuit	Original		REAL[7]			Proposed		
	LC	*LO*	*LC*	*LO*	*Key*	*LC*	*LO*	*Key*
C432	9	21	0	31	1	9	13	1
C499	40	130	15	98	5	15	98	5
C880	49	47	0	46	5	39	35	5
C1355	112	346	0	296	15	21	262	15
C1908	104	449	16	507	30	53	334	30
C2670	49	686	0	716	20	16	609	20
C3540	288	1139	0	1171	30	76	923	30
C5315	69	1285	1	1227	35	2	1161	35
C7552	214	2212	54	2205	50	97	2078	50

TABLE V. SIGNALS WITH LC AND LO BY INCREASING THE KEY LENGTH

Circuit	Original		REAL[7]			Proposed		
	LC	*LO*	*LC*	*LO*	*Key*	*LC*	*LO*	*Key*
C432	9	21	0	31	6	5	11	6
C499	40	130	0	98	8	0	98	8
C880	49	47	0	46	14	39	21	14
C1355	112	346	0	296	40	0	223	40
C1908	104	449	16	507	50	53	266	50
C2670	49	686	0	716	46	10	526	46
C3540	288	1139	0	1171	50	61	899	50
C5315	69	1285	0	1225	50	1	1020	50
C7552	214	2212	54	2205	50	97	2078	50

V. CONCLUSION

In this paper, a new logic encryption method is presented that in addition to encryption, reduces the number of signals with common vulnerability factors. It also significantly eliminates signals with low controllability and low observability. This work will largely prevent trojan insertion. To prevent SAT attacks, the encryption key is generated using the output of PUF. Based on the output of PUF, each chip will have a unique encryption, which it makes counterfeiting, IP piracy, and reverse engineering harder.

REFERENCES

[1] Guin, Ujjwal, Domenic Forte, and Mohammad Tehranipoor. "Anti-counterfeit techniques: from design to resign." In Microprocessor Test and Verification (MTV), 2013 14th International Workshop on, pp. 89-94. IEEE, 2013.

TABLE VI. COMPARATION BETWEEN PRESENT METHOD AND [7]

Circuit	SOth = 0.08								SOth = 0.09								SOth = 0.1							
	Original		REAL[7]			Proposed			Original		REAL[7]			Proposed			Original		REAL[7]			Proposed		
	LC	*LO*	*LC*	*LO*	*Key*	*LC*	*LO*	*Key*	*LC*	*LO*	*LC*	*LO*	*Key*	*LC*	*LO*	*Key*	*LC*	*LO*	*LC*	*LO*	*Key*	*LC*	*LO*	*Key*
C432	9	34	0	44	6	6	12	6	9	37	0	47	5	6	26	5	9	39	0	49	1	8	36	1
C499	40	130	0	98	8	0	98	8	40	130	0	98	8	0	98	8	40	130	0	106	8	0	106	8
C880	49	49	0	58	11	21	30	11	49	70	0	74	10	41	38	10	49	77	0	95	18	29	49	18
C1355	112	346	0	302	32	0	266	32	112	346	0	302	40	0	248	40	112	346	0	320	28	0	264	28
C1908	104	474	16	527	50	53	311	50	104	489	16	553	50	53	326	50	104	500	16	564	50	53	341	50
C2670	49	737	0	764	50	8	577	50	49	744	0	775	48	8	591	48	49	706	0	797	43	6	626	43
C3540	288	1227	0	1244	50	74	956	50	288	1258	0	1280	50	75	1021	50	288	1281	0	1298	50	74	1031	50
C5315	69	1371	0	1315	50	4	1100	50	69	1406	0	1348	50	6	1199	50	69	1452	0	1400	50	0	1237	50
C7552	214	2272	54	2277	50	98	2155	50	214	2310	54	2319	50	19	2187	50	214	2358	54	2366	50	96	2336	50

[2] Chakraborty, Rajat Subhra, and Swarup Bhunia. "Hardware protection and authentication through netlist level obfuscation." In Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, pp. 674-677. IEEE Press, 2008.

[3] Roy, Jarrod A., Farinaz Koushanfar, and Igor L. Markov. "EPIC: Ending piracy of integrated circuits." In Proceedings of the conference on Design, automation and test in Europe, pp. 1069-1074. ACM, 2008.

[4] Rajendran, Jeyavijayan, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. "Logic encryption: A fault analysis perspective." In Proceedings of the Conference on Design, Automation and Test in Europe, pp. 953-958. EDA Consortium, 2012.

[5] Dupuis, Sophie, Papa-Sidi Ba, Giorgio Di Natale, Marie-Lise Flottes, and Bruno Rouzeyre. "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans." In On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International, pp. 49-54. IEEE, 2014

[6] Baumgarten, Alex, Akhilesh Tyagi, and Joseph Zambreno. "Preventing IC piracy using reconfigurable logic barriers." IEEE Design & Test of Computers 27, no. 1 (2010).

[7] Rathor, Vijaypal Singh, Bharat Garg, and G. K. Sharma. "A Novel Low Complexity Logic Encryption Technique for Design-for-Trust." IEEE Transactions on Emerging Topics in Computing (2018)

[8] Subramanyan, Pramod, Sayak Ray, and Sharad Malik. "Evaluating the security of logic encryption algorithms." In Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on, pp. 137-143. IEEE, 2015.

[9] Rajendran, Jeyavijayan, Huan Zhang, Chi Zhang, Garrett S. Rose, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. "Fault analysis-based logic encryption." IEEE Transactions on computers 64, no. 2 (2015): 410-424.

[10] Yasin, Muhammad, Jeyavijayan JV Rajendran, Ozgur Sinanoglu, and Ramesh Karri. "On improving the security of logic locking." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35, no. 9 (2016): 1411-1424.

[11] Liu, Bao, and Brandon Wang. "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks." In Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, pp. 1-6. IEEE, 2014.

[12] Kamali, Hadi Mardani, Kimia Zamiri Azar, Kris Gaj, Houman Homayoun, and Avesta Sasan. "LUT-Lock: A Novel LUT-based Logic Obfuscation for FPGA-Bitstream and ASIC-Hardware Protection." arXiv preprint arXiv:1804.11275 (2018).

[13] Yasin, Muhammad, Bodhisatwa Mazumdar, Ozgur Sinanoglu, and Jeyavijayan Rajendran. "Security analysis of anti-sat." In Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific, pp. 342-347. IEEE, 2017.

[14] Jain, Sunil K., and Vishwani D. Agrawal. "Statistical fault analysis." IEEE Design & Test of Computers 2, no. 1 (1985): 38-44.

[15] Mehdi Ayat, Reza Ebrahimi Atani, Sattar Mirzakuchaki, “On Design OF PUF-Based Random Number Generators”, International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.3, May 2011, Pages 30-40.