Attacks on locked Combinational and Sequential circuits

A M. Tech Project Report Submitted
in Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Subham Das
(204101056)

under the guidance of

Dr. Chandan Karfa

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Attacks on locked Combinational and Sequential circuits**" is a bonafide work of **Subham Das** (204101056), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Chandan Karfa**

Associate Professor

June, 2024

Department of Computer Science & Engineering

Guwahati.

Indian Institute of Technology Guwahati, Assam

Abstract

With the advancement of time, several logic locking techniques both for combinational and sequential circuits are proposed to protect IC from many hardware security issues. On the same time, attacks against such obfuscation techniques are also devised to overthrow the obfuscation methods. One of the objective of this work is to test the effectiveness of such an attack upon a state-of-the-art defence mechanism for combinational circuits. Other objective is to formulate attacks against locked sequential circuits.

Acknowledgements

First of all, I would like to express my sincere gratitude and regards to my MTP supervisor Dr. Chandan Karfa, Department of Computer Science and Engineering, IIT Guwahati for his encouragement, guidance and constant support. His excellent advice and efforts helped me to learn several useful and invaluable lessons, which helped me to do this project work and also would encourage me in future do perform well.

I would like to thank Sarthak Behera, an UG student, IIT Delhi for helping me understand certain parts wherever I got stuck.

Also, I would like to thank Praveen Karmakar, Research Scholar in Computer Science and Engineering Department, for the support he has given.

Last but not the least, I would like to thank my parents and my brother for giving me constant motivation and support. They taught me great lessons and are the most significant support in my life.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

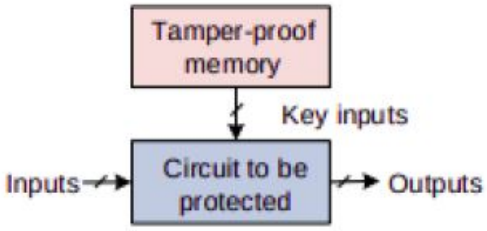
The manufacturing costs of the integrated circuits(ICs) have increased, and has led to an era of fabless semiconductor companies. The semiconductor companies are investing their most of the money on research and are offshoring third-party foundries to manufacture their chips. But this leads to a different issue.As the third-party foundries have access to the chip layout, and if the foundries are untrustworthy then this outsourcing may result in huge loss for the semiconductor companies instead of cost saving by lowering labour and manufacturing cost. There can be different hardware security vulnerabilities such as IP piracy, overbuilding, reverse engineering and hardware trojans. Also, the law enforcement of IP protection vary from country to country and thus the need to hide the functionality of the original circuits from the untrustworthy foundries. There is a solution for overcoming this issue. It is known as logic locking.

1.1 Logic Locking

Logic locking techniques [8], [9], [10], [11], [12], are designed to overcome such threats. The basic idea is to introduce additional gates which are termed as key gates during the IC design. These key gates will be controlled by additional inputs known as keys and the circuit will function properly only when the correct keys are provided. Also at the same

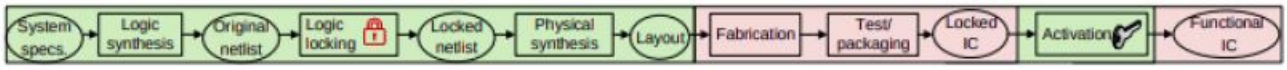
time, the circuit must act erroneously whenever wrong keys are being applied. An abstract view of logic locking is shown in Fig.1.1.

Fig. 1.1 Abstract representation of logic locking



The keys are stored in a tamper proof memory so that it remains safe. The keys are loaded into the memory at the time of activation of the chip. This way the circuit is locked using some logic locking technique. Thus, even if the untrustworthy foundries have access to the locked circuit net-list, they will not be able to get functional circuit without performing processes like reverse engineering [13]. The whole IC design flow involving locking and IC activation is shown in Fig.1.2.

Fig. 1.2 Design flow for locking and activation of an IC

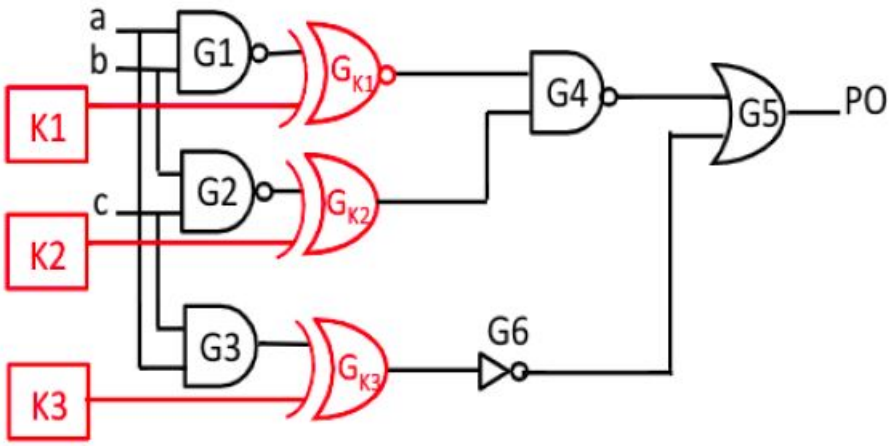


So, the secret key added to the circuit is only known to the designers and all the other untrustworthy foundries can fabricate the chip without knowing this key. Thus, the purpose of protection from IP theft is served very well. For achieving this, additional logic gates termed as key gates are inserted into the circuitry. In general, XOR/XNOR gates are used for serving the purpose. At the end, a locked net-list is obtained which useless without the secret key.

An example of a locked circuit is shown in Fig.1.3. The circuitry without the red part is the original net-list. To protect it using the logic locking technique, additional key gates G_{k1}, G_{k2} and G_{k3} which are controlled by K1, k2, and K3 respectively. The key gate G_{k1}

acts as a buffer for K1 = 1 and, acts as an inverter for K1 = 0. Whereas, key gate G_{k2} acts a buffer for K2 = 0, and as an inverter for K2 = 1. Finally, as there is a NOT gate G6, in front of key gate G_{k3}, it acts a buffer when K3 = 1, and acts as an inverter when K3 = 0. The key gates just acts as a buffer in the locked circuit, as removal of those will give us the original net-list. Thus, the correct key values are **K1 = 1, K2 = 0, K3 = 1**.

Fig. 1.3 A locked circuit, locked using key gates(XOR/XNOR). The correct values of keys K1, K2, K3 are 1, 0, 1 respectively.



This is the basic idea of logic locking devised for protecting the hardware. But this is not enough as several attacks are made on the locked circuits and are successful in unlocking them. So, now different defence mechanisms are being proposed and similarly attacks upon those defence mechanisms are also being made.

1.2 Motivation

1.2.1 Combinational Locking

In the previous section we saw basic logic locking technique which is done at the gate-level. In [1], they have proposed an idea of locking by raising the abstraction level, i.e., by locking the circuit at the register-transfer level(RTL). The key idea of this proposal is to obfuscate a design at the algorithm-level so that the obfuscation is semantically meaningful. So, the

C/C++ code is locked and after the high-level synthesis a locked RTL is obtained and this RTL is logically synthesised to obtain a locked net-list. [2] talks about an attack strategy upon the RTL locking technique proposed in [1]. But the algorithm for attack given in [2] is getting timed out for the larger benchmarks locked with TAO [1]. So, Dr. Chandan Karfa, Associate professor of IIT Guwahati and Sarthak Behera, an UG student of IIT Delhi has come up with a modified approach based upon the approach given in [2].

1.2.2 Sequential Logic Locking

The squential circuits are also locked using various techniques. A sequential circuit has two parts, sequential component comprising of flip-flops and register elements and data path comprising of combinational logic gates. For locking of sequential circuits, mainly two strategies are being used.

- Locking only the data path containing the combinational gates [3].
- Locking the sequential component [4].

The attacks on sequential circuits where combinational parts are locked is done by performing attacks on individual combinational blocks using the help of scan chain. To counter this attack method, in [3], the scan chain is made unavailable to the attacker using Dishonest oracle. Whenever the attacker tries to enable the scan chain bit, the oracle will produce corrupted output, thus making the oracle dishonest. As a result complete circuit needs to be taken into consideration.

Another methodology is given where [4] [5] sequential part of the circuit is locked by adding additional states using flip-flops. For the circuit to produce correct outputs, an input sequence has to be given, otherwise the circuit malfunctions.

The time taken to break a sequential circuit using SAT based attacks is much more as compared to combinational circuits. There has been some recent works for reduction of time consumed by SAT based attacks on locked sequential circuits, [14] with incremen-

tal bounded model-checking, and dynamic simplification of key-conditions (Key-Condition Crunching or KC2), using which the obfuscation time has reduced two orders of magnitude. Another work [15] is done where the SAT attack is performed on the locked sequential circuit without the scan access, by finding discriminating set of input sequences which is sufficient to determine the functionality of the locked gates. Though novel ideas are used to reduce the attack time, both these SAT based attacks are not scalable for large circuits.

Therefore, an new attack methodology apart from SAT based attacks is required to de-obfuscate the sequential locked circuits where combinational parts of the circuit is being locked.

Also, an attack methodology needs to be devised to counter the sequential locked circuits where sequential parts of the circuit is being locked.

1.3 Objective

1.3.1 Combinational Logic Locking

- Create several test cases for the given benchmarks - WAKA, ARF and Motion, which are locked using TAO [1], and check the effectiveness of the improved attack strategy on the defence mechanism TAO [1].
- Also create several test cases of locked C code and test the attack strategy on them and report the analysis.

1.3.2 Sequential Logic Locking

- To study whether key information for locked sequential circuits where combinational parts are locked [3] can be revealed by designing an attack using Welch t-test [16].
- To design a structural analysis based attack methodology for countering the sequential locked circuits where sequential part of the circuit is being locked [4] [5].

1.4 Chapter Organizations

The rest of the report is organised as follows: Chapter 2 contains three sections - Combinational Logic Locking and Sequential Logic Locking and Welch-T test. The first section consists of the explanation of the defence techniques proposed in [1] and the explanation of the attack strategy on [1] proposed in [2]. Then it discusses about the locking mechanisms for sequential circuits. The next section of Chapter 2 consists of basic idea of Welch-T test based attack proposed in [16]. Chapter 3 contains the explanation of the improved attack technique on [1] and it's implementation details along with changes made with respect to the existing algorithm in [2]. Then it discusses regarding the attack techniques on Dishonest Oracle [3] and Harpoon [4] [5]. The results of the experiments using Modified SMT attack on TAO [1] and attacks on the locked sequential circuit will be presented in Chapter 4. Finally, the report will be concluded in Chapter 5.

Chapter 2

Literature Survey

There has been a lot of research in the filed of hardware obfuscation techniques, especially on logic locking mechanisms as well as on the attack strategies for the defence mechanisms. In [17], [18] the evolution from start to current scenario are being discussed.

2.1 Combinational Logic Locking

Here, we will discuss the techniques used in [1] for obfuscation the circuit at the algorithm level. And also discuss the attack technique in [2] and it's limitations.

2.1.1 TAO:Techniques for Algorithm-level Obfuscation [1]

An algorithm can be characterised by several elements to be protected, they are: Constants, Control flow, Operations and Dependencies.

- **Constant Obfuscation**: Constants represent hard-coded values used by the algorithm (coefficients, thresholds, etc). Such m-bit constants can be locked using a m-bit key value by doing XOR operation to form a m-bit locked constant value. This locked constant value will be given in the circuit and the key value will be put in the tamper proof memory during activation of the chip. Such that,when again a XOR operation

is being performed between the key value present in the tamper proof memory and the locked constant value, it will produce the actual correct constant value.

- **Control-Flow Obfuscation**: Control-flow branches describe the progress of the algorithm. The condition for a branch statement can be masked with a 1-bit key. Suppose for a condition C, when C=1, statement_1 is executed and when C=0, statement_2 is executed. Now, let's say the condition is being masked with a 1-bit key, K=1. Thus, XOR of C and K, will invert the value of C, and therefore the statements are swapped with each other in the locked code along with K=1 stored in tamper proof memory. Now, without the correct value of the key, the untrustworthy foundry will not be able to get which flow to be taken when the condition is false/true.

- **Operation Obfuscation**: Operations are the computational part of the algorithm. Operations can me camouflaged using a 1-bit key. A fake operation can be formed and associated with the actual operation. The key value decides which is the correct operation. Suppose, there is an operation $a = b + c$.It can be obfuscated by associating a fake operation with it, such as $a = b - c$, and write the statement using ternary operator as $a = key\ ?\ b + c : b - c$, where key = 1. Now, it is difficult to know the correct operation by any untrustworthy foundry without the knowledge of the correct key value.

- **Dependence Obfuscation**: Operation dependence describes how the data values are used. Different variants of a dependence can be formed using a k-bit key. With k-bit key, 2^k variants are possible, and correct paths are being activated once the chip comes from the fabrication lab. It will be difficult for the untrustworthy foundry to get hold of the actual variant of the dependence without the correct key value.

2.1.2 Analysis of the TAO obfuscation technique

- For control flow obfuscation and constant obfuscation the area overhead is minimal. Only a XOR/XNOR gate is required for the implementation of the obfuscation techniques. These obfuscations are easy to apply.
- In case of operation obfuscation, an additional fake operation has to be added. Along with it a multiplexer is required which would select one of the operation based on the key value. Thus, this obfuscation incurs a reasonable overhead. This technique is also easy to apply.
- Dependence obfuscation is a very powerful technique. It creates several semantically different but feasible code variants. This technique incurs a significant area overhead as many additional operators and connections/multiplexers are required.
- [1] states two solutions for key management, key folding and AES-based architecture.

2.1.3 SMT based attack on register transfer level locking [2]

The attack is an oracle guided attack, and is motivated from [19] In oracle guided attacks, the attacker uses the unlocked IC, bought from the market and use it for obtaining the input-output pairs. Conventional gate-level attacks are not scalable and TAO [1] applies defence techniques that locks the circuit during high level synthesis, thus the resulting locked net-lists are consequently less vulnerable to conventional gate level attacks. So, an attack working at higher level of abstraction is desirable.

Satisfiability Modulo Theories (SMT) provide a much better modelling capability than the Boolean Satisfiability (SAT), by adding equality reasoning, arithmetic, and other useful first-order theories. Hence, SMT [20] is more suitable for higher abstraction level.

To attack designs locked using TAO [1], the functionality needs to be abstracted from the RTL Design. So, the TAO [1] locked designs are converted into an FSMD model [21]. In [2] the high level behaviour called RTL-FSMD is extracted from the locked RTL

using the rewriting method [22]. The algorithm proposed will find distinguishing input patterns(DIPs) in an iterative fashion, and rules out equivalence classes of incorrect keys in each iteration. The algoriothm halts when there are no DIPs found.

Distinguishing Input Pattern: An input pattern which gives different output pattern for two different key input pattern.

- **Attack flow**

  The clause, such that for a given input pattern I^d, with two different key input patterns K_1 and K_2, we get different outputs O_1 and 0_2, is given as $P(I^d,O_1,K_1) \wedge P(I^d,O_2,K_2) \wedge (O_1 \neq O_2)$. This clause is solved using Z3 SMT solver [23], and if satisfiable, gives a DIP. After getting the DIP, the inputs are evaluated on the unlocked circuit and the key inconsistent with the I/O pair being eliminated. New clauses are being added to the Z3 SMT solver [23], thereby eliminating an equivalence class of keys which is wrong instead of eliminating just a single key. Again the algorithm tries to find more DIPs based on the new conditions, added to the SMT solver. This process goes until there are no DIPs reported. The algorithm iteratively eliminates a group of wrong keys and at the end is left with only the correct key. Thus, the correct key is returned by the algorithm.

- Analysis of the attack proposed by [2] on the TAO The attack tool as tested on three HLS Benchmarks - WAKA, ARF and Motion. The attack is successful in most of the benchmarks For the successful cases, the time to break the circuit is under 30 minutes. The timeout was set to 10 hours. Few of the test cases were timed out.

2.2 Sequential Logic Locking

The attack methodology on locked sequential circuits using oracle, assumes that scan chain access is available to the attacker. An attacker can mount SAT based attack on individual combinational blocks by leveraging scan chain access. Specific values to registers can be

set using scan chain to get correct input-output pairs for individual combinational blocks. In response to this attack mechanism, there are obfuscation techniques [24] [3] which tires to disable the scan chain access. Thus, the attacker will obtain incorrect output if tried to access the scan chain.

In Fig. 2.1, it is shown that the key is fed to logic locked design and the key will be correct or incorrect is chosen based upon the output of the D Flip-Flop. Scan chain access will lead scan enable to be set and in turn the select line to the multiplexer will be true and wrong key will be fed to the logic locked design. So, the outputs obtained when scan-enable is true will wrong, therefore corrupting the oracle during scan chain access.

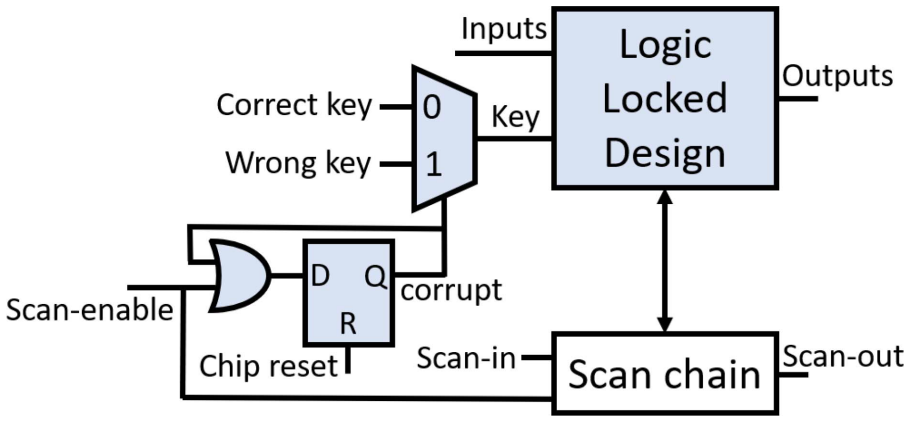


Fig. 2.1 Conceptual architecture of Dishonest Oracle [3]

The flow of a sequential circuit can be defined using a finite state machine (FSM). This FSM can be locked [4] by addition of new FSM states, which will be spurious states. In FSMs, based upon an input the transition from the current state to the next state is being decided. Additional states are being added before the actual correct reset state. Therefore, a specific input sequence from the new reset state, which is a spurious one, will lead to the correct reset state of the circuit. This input sequence is defined as the key for the locked FSM. So, the circuit works only when the correct input sequence is being applied. The key is added implicitly. Fig. 2.2 depicts the FSM locking, where additional states S0, S1, S2 and S3 are added as spurious states and remaining states are the part of the original FSM. The circuit will work properly onlt when the control reaches in S4 in Fig. 2.2, and to

reach S4, a sequence of inputs need to be given at the beginning. This sequence of inputs is considered as key for the obfuscated sequential circuit.

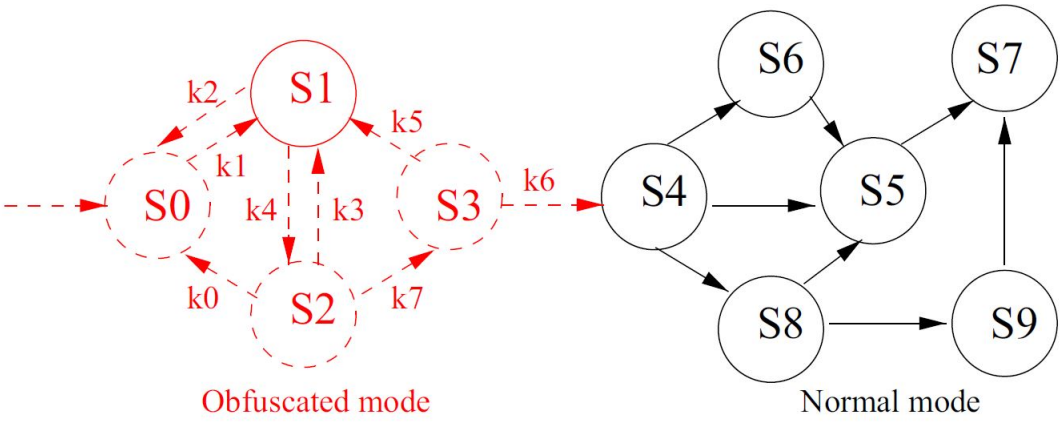
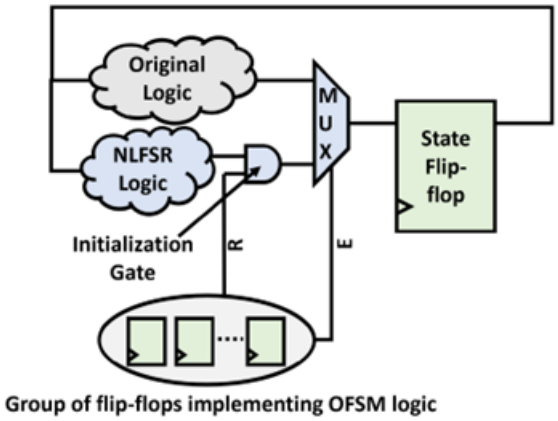


Fig. 2.2 FSM Locking [6]

A practical implementation of the idea in [4] is presented in [5], where multiplexers are used for obfuscating the original FSM design. The In Fig. 2.3, we can see the original logic of the circuit is obfuscated using group of flip-flips which belongs to obfuscated FSM, and its output is the select line for a multiplexer which has two inputs one is the original circuit and another one is the NLFSR logic. So, here we can see that the obfuscation is being done without hampering the original circuit.

Fig. 2.3 Implementation of FSM locking



2.3 Welch T-test

The Welch t-test is used to compare the means of two groups of samples when the variances are different. Let the two groups be a and b. The formula is:

$$t = \frac{m_a - m_b}{\sqrt{\frac{v_a}{n_a} + \frac{v_b}{n_b}}}$$

where m_a and m_b are mean of group a and b respectively, v_a and v_b are variance of group a and b respectively, and n_a and n_b are number of samples in group a and b respectively. Welch-t-test computes a t-value. If the absolute value of the t statistic ($|t|$) is greater than the critical value, then the difference is significant. Else it is not.

Critical value for t statistic ($|t|$): The Welch's t-test was used in Test Vector Leakage Assessment (TVLA) [25] aims at being able to provide detection of information leakage using statistical analysis. It is seen that a threshold of 4.5 standard deviations was set as critical value for using this test and the same is used in most scenarios. If for a given test, t statistic ($|t|$)) exceeds ± 4.5, then there is significant different in the two groups on which the test is applied.

2.3.1 Welch t-test as an attack model

The statistical method is utilised as an attacking mechanism for combinational logic locked circuit in [16]. The basic idea is to determine how two groups of output vectors are different from each other. Random input vectors and key vectors are being chosen and two set of output vectors are obtained, one from the oracle which will be giving the correct outputs for the given input and another is the from the locked circuit with randomly chosen key input vectors. Welch t-test is being performed among these two groups and t-values are being obtained output bit wise. The key vectors for which any one of the output bit has t-value $\geq |4.5|$ are taken and formed a group. Two thresholds values are defined as :

- Output Consideration Threshold : A percentage value used for filtering the highly affected output values for one set of experiment. The output bit is considered as highly affected only if percentage of high t-value for the output bit is more than the threshold value.

- Key Consideration Threshold : A percentage value used for filtering out key bits based upon occurrence of a value for a key bit.

Therefore, based upon the Output Consideration Threshold output bits are filtered and keys affecting those output bits are considered. The key bits are then filtered based upon Key Consideration Threshold and the key value is flipped for the filtered key bits. These, key bits which are being flipped are cnsidered as newly found keys values.

2.4 Conclusion

This chapter discussed about a particular obfuscation technique on combinational circuits called TAO [1] and an attack [2] upon the circuits locked by TAO. The attack is effective in breaking the obfuscation.

It also discusses about the obfuscation techniques [4] [5] [3] used for locking the sequential circuits.

Finally, a statistical attack model [16] for attacking logic locked circuits just based upon input output co-relation, is discussed.

Chapter 3

Attacks on locked combinational and sequential circuits

3.1 Modified SMT attack on TAO [1]

The baseline SMT attack [2] gave time-out for some of the benchmarks locked using TAO [1]. In the baseline attack [2], the checking for UNSAT of the DIP formula at the last iteration of the attack after addition of enough constraints takes a lot of time and thus times out eventually. One solution can be, addition of time-out value in the Z3 SMT solver [23]. But there is an issue here, it's difficult to set an ideal time out value for the SMT solver, as when there will be a time-out it does not guarantee that it was the last iteration i.e., no more incorrect key is remaining in the key search space.

In the baseline attack [2] the last iteration was getting the same key input assigned in the SMT formulation and thus Z3 solver was unable to find a DIP and eventually got stuck. So, a new formulation with a constraint that keys to be assigned cannot be equal was added along with a timeout. This new idea is termed as Modified SMT Attack or Modified TAO Break. After the solver times out, all the keys remaining in the key search space be extracted and a set of probable keys be created. However, if the remaining keys are very

large it would incur unnecessary time on finding them, this also needs to be handled using a threshold value. After, the remaining key set is obtained, SMT attack on the finite key set to be performed and eliminate wrong keys and return the correct key at the end of the iteration.

Combining the above mentioned ideas, the attack strategies are as follows:

- Set the initial time-out value and the threshold size for the remaining key set.
- Run the modified SMT attack with the initial time-out value as the core method. .
- When the algorithm times out, find the remaining keys. If the remaining keys are beyond the threshold value, increase the timeout value and again run the algorithm.
- If the size of the remaining key set is less than the threshold value, run the SMT attack on the remaining key set for obtaining the correct key.

The tool flow is given in Fig.3.1

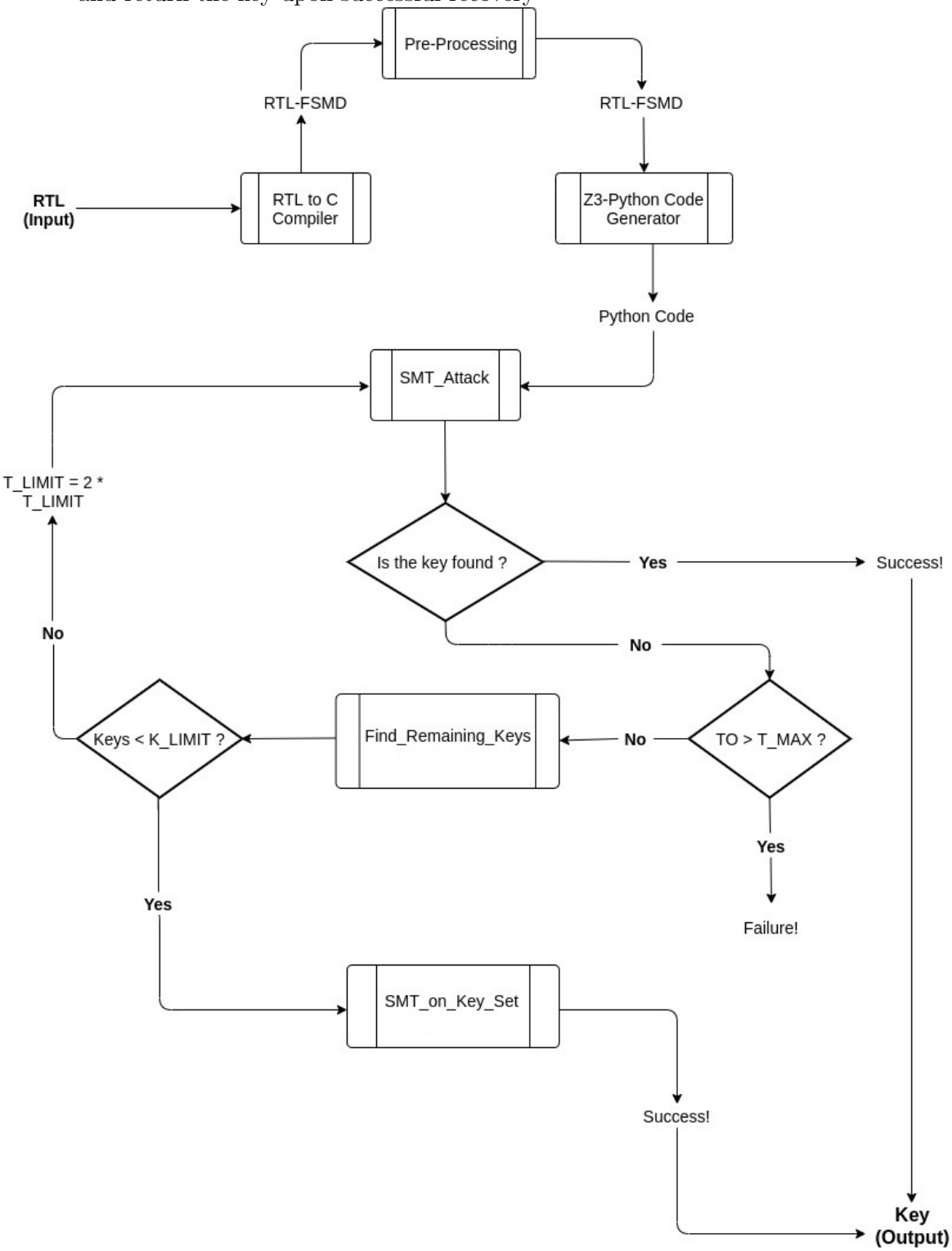
3.1.1 Effectiveness of the modified SMT attack on TAO [1]

Multiple test cases for the HLS benchmarks WAKA, Motion and ARF is made with all the locking techniques given in TAO [1] for checking the robustness of the modified TAO Break algorithm. The detailed results of the experiment on the benchmarks are given in Chapter 4.

3.2 Attack on locked sequential circuit, locked using Dishonest Oracle [3]

Our main idea is to utilize the Welch t-test attack on combinational circuits [16] with minor modifications, for de-obfuscating sequential locked circuits. The locking is done in the combinational parts of the sequential circuits.

Fig. 3.1 Scalable SMT Attack tool flow: The framework takes an RTL input and return the key upon successful recovery



3.2.1 Circuit latency identification

The attack using Welch t-test in [16] is performed upon combinational circuits, whereas here we are attempting to use the same attack model on sequential circuits. The standard behaviour of the oracle for combinational circuits is that an input is given and the desired output is obtained in the next clock, which is the correct output for the given input. In case of sequential circuits correct output generation for a given input is different. For sequential circuits, there is an output at the end of every clock cycle. Therefore, we have an oracle for a sequential circuit with us but do not have the value for the number of cycles required to get the correct output. So, here the challenge is to determine the clock period of the circuit, because without the correct clock period the oracle is of no use to us as we will be unable to obtain the correct output for any input pattern, and without correct input-output pair the attack cannot be applied.

Now, for determining the clock period of the sequential circuit, we came up with an idea to count the number of DFFs in a path from every input to all the outputs reachable. Here we are assuming for initial phase of experiment that the logic locking scheme applied is not affecting the latency of the circuit, the internal state values are not being considered in this idea.

Methodology

Our idea is to get the count of number of DFFs in a path from every input to a given output and then considering the maximum of those values as the latency for obtaining the correct value for the given output bit. In similar manner, the latency was computed for all the output bits in the circuit. Finally, the maximum of all the latency values of the output bit was considered as the latency of the whole circuit.

l_{11} = Number of DFFs from $input_1$ to $output_1$

l_{21} = Number of DFFs from $input_2$ to $output_1$

...

$$l_{n1} = \text{Number of DFFs from input}_n \text{ to output}_1$$

$$\text{Latency for output}_1,\ L_1 = \text{maximum } (1_{11},\ l_{21},\ ...,\ l_{n1})$$
$$\text{Latency for output}_2,\ L_2 = \text{maximum } (1_{12},\ l_{22},\ ...,\ l_{n2})$$
$$...$$
$$\text{Latency for output}_m,\ L_m = \text{maximum } (1_{1m},\ l_{2m},\ ...,\ l_{nm})$$

$$\text{Latency of the circuit, } L_{cir} = \text{maximum } (L_1,\ L_2,\ ...,\ L_m)$$

Implementation

The inputs, outputs, keys and gates of the circuit are considered as nodes and wires as paths. Two different implementations were done, one with Depth-First Search and another with Breadth-First Search. Unfortunately, there were problems with both the implementations. Depth-First Search is taking a long time and thus is not scalable. For Breadth-First Search results are inaccurate.

In the Fig 3.2, it is shown that there are two paths from Input 1 to Output 1. Now, ideally the latency should be the maximum of both the paths, which is 4. But if BFS is applied it comes as 3, because first iteration Node a and Node d is visited. Then in second iteration, Node b and Node c will be visited, finally it will visit Output 1. Since, Node c will be visited by Node d first, Node b cannot visit Node c again, thereby giving wrong results.

So, at the end we went for an heuristic approach by manually running the benchmarks with random clock values. The benchmarks were giving all 0's in the output for most of the clock values, except the *medium*. And thus, we proceeded with the Welch t-test attack on the *medium* benchmark only.

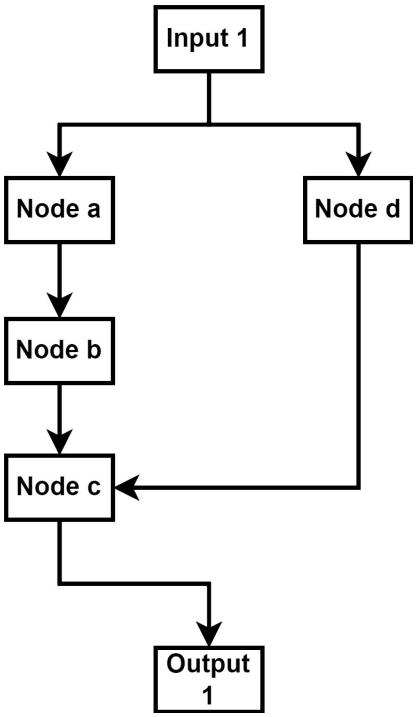


Fig. 3.2 Sample circuit

3.2.2 Attack Methodology

The algorithm given in [16] is modified to perform an attack on sequential circuit. The output based upon latency of the circuit is taken. Apart for regular input values in combinational circuits, an oracle of a sequential circuit takes one clock enable bit and the clock period of the circuit.

The Welch t-test attack was performed upon the *medium* benchmark with the changes in the code and the results are discussed in the results section.

3.2.3 Benchmarks

The benchmarks taken for performing the experiments are from Hardware Logic Locking and Obfuscation: Capture The Flag (HeLLO: CTF), 2021 [7]. For our experiments we have considered the benchmarks under Oracle-Guided Combinational Logic Locking category. The details of the benchmarks are given in table 3.1.

Bench File	Number of Inputs	Number of Outputs	Number of Keys	Number of DFFS
Small	37	70	128	577
Medium	33	22	128	521
Large	33	22	128	941

Table 3.1 Details of the benchmarks given in HeLLO CTF 2021 [7]

3.3 Attack on locked sequential circuit, locked using FSM locking [4]

The idea is to do structural analysis to generate the FSM of the sequential locked circuit given. There are two different approaches for this obfuscation technique [4]:

- Structural analysis to get the obfuscated FSM of the circuit and identifying the correct reset state, the reset state of the original circuit.
- Structural analysis to directly obtain the original FSM of the circuit.

3.3.1 Attack Methodology

We will be doing basic structural analysis of the circuit to extract the behavior of the circuit. The structural analysis will include these two objectives:-

- Identification of the obfuscated FSM

  The key checkpoints for identification of the obfuscated FSM are:

  - Identification of the state registers. State registers are the registers which defines the transitions and aids in formation of the FSM.
  - Identification of the correct pairs (next state – current state) of state registers from the group of identified state registers.
  - Formation of the obfuscated FSM using the transitions obtained from current state and next state values.

- Identification of the correct reset state

  The FSM locking adds additional states to the original FSM as a part of locking scheme. So, basic idea is that the actual states of the original FSM remains untouched, therefore the idea is to identify the correct reset state from the states of the obfuscated FSM.

Two different approaches will be discussed here.

Methodology 1

- **Identification of next state – current state pairs**

  An observation is made by analyzing the original circuit of the warmup benchmark for HeLLO CTF 2022 [26], is that the next state register's output, i.e. next state, is the input of the current state register. This implies the next state register - current state register pair will have an output to input relationship, i.e. direct edge will be there from the next state register to the current state register. Our approach will be to perform a DFS traversal from all the registers until it reaches another register, and try to get the next state – current state register pairs.

- **Generation of the FSM**

  It is observed that the *next state* is a function of subset of *current states*. Therefore, using all the possible values of the *current state*, the values for *next state* can be obtained. Using the values of *next states* and *current states* the FSM states can be obtained. The states which are non-reachable is discarded and the FSM is drawn using the remaining states.

- **Identification of the correct reset state**

  The idea is to take all the states one by one and set it as the reset state in the circuit and obtain the outputs for random inputs. At the same time using those same inputs

we can obtain the correct outputs using the oracle. Simple comparison technique can reveal the correct reset state.

Methodology 2

The previous de-obfuscation mechanism given in methodology 1 is both structural and oracle-guided. Here, the is another approach where our aim is to extract the original FSM, the FSM before the addition of additional states (also known as spurious states). In the practical implementation of FSM locking in [5], multiplexers are added explicitly for purpose of obfuscating. The work [5] mentions that the input to a state register is multiplexed with another group of logic circuit with a select line which is a output of group of flip-flops which are termed as OFSM logic, which helps in forming the spurious states. So, our idea is to identify the multiplexers in the circuit and analyse the two inputs and a select line of the for getting the group of registers used for different logics like OFSM, NLFSR and original logic.

3.3.2 Implementation

Methodology 1

- **Identification of next state – current state pairs**

  The circuit is considered as a graph where the sequential elements like flip-flops, and combinational elements like AND, OR gates etc are considered as nodes and connections between them as paths. All the registers in the obfuscated circuit are taken and depth first traversal (DFS) is done till it reaches another register. The DFS is taken from the output of a register to the input of another register. The hop count, i.e., number of nodes between two registers is also being computed. Finally, we obtain register pairs and the count of nodes (combinational elements) between them.

If we consider the observation for the original warmup circuit the registers should be directly connected, but since this circuit is being obfuscated, some combinational elements might be added. So, the register pairs having in-between node count as 1 are considered for the next state register - current state register pairs, i.e., only one combinational element between a next state's output and current state's input.

Also, one single register can be considered as two different next state registers, as a register in the circuit is a D Flip-Flop and it can have two outputs Q and $\bar{Q}$. So, Q and $\bar{Q}$ can be two different next state outputs going to different current states.

Another observation made from the original warmup circuit [26], is that the *next state* is dependent on the subset of *current states*. So, we check for whether the identified next states are dependent on the identified current states or not. Now, the identified next state registers are considered and DFS is performed from it's input till it reaches output of another register. For every next state register input, count of identified current state register's output it reaches is found. If a *next state* is not dependent on any of the *current state*, then that next state register is considered to be wrongly identified.

- **Generation of FSM**

  The implementation is given in Algorithm 1

- **Identification of the correct reset state**

  The circuit for the obfuscated circuit can be simulated using verilog code for the circuit. In the verilog code, the reset state of the circuit can be set to all the possible state values of the FSM obtained using Algorithm 1. Therefore, original circuit and modified obfuscated circuit can be compared using Welch - T test method [16]. The predicted reset state will be correct or incorrect, so the T value for correct reset state will be 0. Using this idea we can obtain the correct reset state.

Algorithm 1: Generate_Obfuscated_FSM (Cirucit, Next States set)

Inputs: circuit represented in graph, next states set
Output: Obfuscated FSM
1: **for** all the next states **do**
2: **Depth First Search** from *next state* until it reaches a register and store the operations along the way in the form of a graph *next_state_graph*.
3: **Unroll** the *next_state_graph*.
4: **Topological Sort** of *next_state_graph* from *next state* as output, to form a topological sorted graph *next_state_topo_sorted_graph*
5: Using *next_state_topo_sorted_graph* convert the operations in C code.
6: **end for**
7: Simulate all possible values for *current states* and get the corresponding *next state* values.
8: Using the current state value vectors and the corresponding next state value vectors form another graph *FSM_graph*.
9: Remove the nodes from *FSM_graph* which does not have any incoming edges and update *FSM_graph*

return ***FSM_graph***

Methodology 2

The circuit as a graph is taken and graph analysis is done to identify all the multiplexers in the circuit. Now, only those multiplexers are being selected whose output is an input to a flip-flop. And the output of this flip-flip having input as the output of the multiplexer, will loop back and enter to original FSM and NLFSR logic. Therefore, this flip-flip is a *next state register*.

- **Analysis of the select line of the multiplexer**

  The select lines of the identified multiplexers are taken and DFS is performed from it until it reaches a flip-flip. By referring the circuit in Fig. 2.3, it can concluded that the registers identified in this process belongs to the OFSM logic.

- **Analysis of the inputs of the multiplexers**

  DFS from both the inputs are performed until it reaches a flip-flop. Therefore, we will be getting two groups of flip-flops, NLFSR logic and original logic 2.3.

The NLFSR logic and the OFSM logic is being added to the cirucit later for obfuscating the FSM, so removal of those might leave us with the original FSM. The *current states* will be known to us from the multiplexer output path, so backtracking from these *current states* we can reach the corresponding *next states*. After we get the next state-current state pairs we can use the Algorithm 1 to get the FSM. But time since we will be removing the obfuscated portion of the circuit, we will end up with original FSM or the circuit.

Chapter 4

Results and Analysis

4.1 Resuts for attack on obfuscated Combinational Circuits

Several test cases for the HLS benchmarks - WAKA, ARF and Motion are being formed for the purpose of testing the effectiveness of the modified SMT attack on TAO [1]. The results of the test cases are reported in Table 4.1

The approach of TAOBreak is not limited to HLS-generated designs. It also works on locked C code. For proving that it can break the locked C code, several C variants with large number of keys are created and the attack results upon them are reported in Table 4.2.

The results in Tables 4.1 and 4.2 shows that modified TAO Break is effective on breaking the locked circuits as compared to the baseline algorithm [2].

While experimenting with different test cases, an observation was made that the test cases ARF_6 in Table 4.1 and ARF_6 Table 4.2 times out for TAOBreak, while other test cases having larger key size compared to them are broken using the approach. After analysis of the test cases it was found that, whenever the keys depend upon each other, i.e. , for every value of the first key there exists an unique second key value which gives the same output. We have named this as key dependence. Hence, there are multiple correct keys for the obfuscated circuit possible and as the remaining key set size is limited, it always fails

Table 4.1 Results: Unlocking TAO-locked RTL designs.

Benchmarks					Obfuscation				Circuit		Baseline attack [2]		Modified TaoBreak			
	LOC	×	+	-	Operations	Conditions	Constants	Key	Comb	Seq	Iterations	Time(s)	Iterations	Time(s)	Timed out	Rem Key
WAKA	753	-	13	7	-	1	2	65	1255	917	3	6.25	3	0.12	0	0
					9	2	-	11			4	2110.92	3	0.95	0	0
					9	-	-	9			4	9.88	2	0.31	0	0
					7	2	2	73			55	135.72	37	95.64	2	2
ARF	1654	21	27	10	-	3	-	3	19715	3381	-	TO	2	2.46	0	0
					-	-	2	64			2	1.09	2	2.24	0	0
					32	3	-	35			-	TO	6	130.82	3	1
					32	3	5	195			-	TO	6	237.83	3	0
					35	3	7	262			-	TO	9	1587.89	6	3
					35	3	8	294			-	TO	-	TO	-	-
					37	3	9	328			-	TO	8	5207.75	8	3
MOTION	1250	15	29	10	-	-	2	64	13938	2924	2	0.91	2	0.34	0	0
					27	-	-	27			-	TO	2	4.82	0	1
					27	-	4	155			-	TO	5	57.37	2	2
					28	-	7	252			-	TO	4	419.04	6	3
					29	-	8	285			-	TO	5	416.84	5	2
					32	-	9	320			-	TO	8	2432.21	8	4

LOC: # of lines in Verilog RTL. ×: # of multiplications in input C. +: # of adds in input C. -: # of subtracts in Input C. Operations: # of operations obfuscated. Conditions: # of conditions obfuscated. Constants: # of constants obfuscated. Key: # of key bits. Comb: # of combinational cells. Seq: # of sequential cells. Iterations: # of iterations. Timed out: # of times SMT solver timed out Rem key: # of times remaining key is computed

to break the obfuscated circuit. The algorithm is not able to generate DIPs after certain iterations for such test cases having key-dependence. One explanation can be that the Z3 SMT solver [23] is having difficulty in showing UNSAT due to the scenario of multiple correct keys.

Example of a key-dependence scenario

Table 4.2 Results: Unlocking a locked C code

	Obfuscation				Baseline Attack [2]		Modified TAO Break	
Bench	Operations	Conditions	Constants	key	Iterations	Time (s)	Iterations	Time (s)
WAKA	1	1	5	162	6	7.92	6	2.72
	3	1	7	228	8	13.38	6	2.94
	5	2	8	263	9	233.58	7	2.29
ARF	2	1	1	35	-	TO	2	2.96
	-	-	4	128	1	1.29	1	0.77
	2	1	2	67	-	TO	3	4.07
	11	3	5	174	-	TO	4	9.56
	15	3	8	274	-	TO	3	118.40
	17	3	9	308	-	TO	-	TO
MOTION	2	-	2	66	2	0.36	4	0.44
	6	-	6	198	-	TO	7	3.09
	12	-	8	268	-	TO	6	27.30
	12	-	10	332	-	TO	11	83.80
	14	-	12	398	-	TO	16	759.93

Consider constant obfuscation as per TAO [1] is applied upon two constants associated with two intermediate variables, say *var_1* and *var_2*. Let, the keys be *key_1* and *key_2*. Assume there is an output variable within the code, say *out*, whose value is computed using the values of *var_1* and *var_2*. There is a condition that both the obfuscated constants have to be associated with same operation with their respective variables. Say, if one is added to *var_1* and other is multiplied to the *var_2* will not lead to key dependence. Now, though the constants associated with *var_1* and *var_2* are obfuscated independently, the keys get dependent upon each other through the output variable, *out*. In locked RTL, *out* = input_1 + input_2 + key_1 + key_2. Therefore, for any value of *key_1* there will always exist an unique value of *key_2* which will unlock the obfuscated circuit. The pseudo-code for the example is given below in listing 4.1 and the locked RTL for the corresponding input C code is given in listing 4.2

Listing 4.1 Input C-code

```
var_1 = input_1 + 10
var_2 = input_2 + 5
out = var_1 - var_2
```

Listing 4.2 locked RTL code

```
reg_1 = input_1 + key_1
reg_2 = input_2 + key_2
out = reg_1 - reg_2
```

4.2 Results for attack on obfuscated Sequential Circuits

4.2.1 Results for attack on Dishonest Oracle [3]

The modified Welch-T test was ran only on the *medium* benchmark, due to lack of knowledge of latency of the circuit. The results are given in table 4.3 . Most of the keys obtained from the Welch t-test attack on the benchmark is wrong. So, further insights and improvements are required for reducing the count of wrong keys.

Bench File	Number of Keys	Number of Keys found	Number of keys correct
Small	128	-	-
Medium	128	113	54
Large	128	-	-

Table 4.3 Resuts for attack on benchmarks given in [7]

These results were presented in a global logic locking competition, *Attacks on Hardware Logic Locking & Obfuscation Capture The Flag 2021* [7] and our team *Team_IITG* from Indian Institute of Technology, Guwahati secured 3^{rd} position [27].

4.2.2 Results for attack on Harpoon [4] [5]

Using Methodology 1 in Section 3.3.1

For the warmup benchmark both the original and obfuscated circuit was given. Other benchmarks - low, medium and high only obfuscated circuits were given. The current state-next state pairs for the obfuscated warmup and low benchamrks are found and are given in Tables 4.4 and 4.5 respectively.

The FSM for the original circuit was easily formed using the Algorithm 1 and the FSM is given in Fig. 4.1. In similar manner, the FSM for the obfuscated circuit for low benchmark was formed and is shown in Fig. 4.2.

We were not successful in running the obfuscated circuit by setting a reset value to the circuit and so could not produce results for identification of the correct reset state of the circuit.

Fig. 4.1 Original FSM of the warmup benchmark

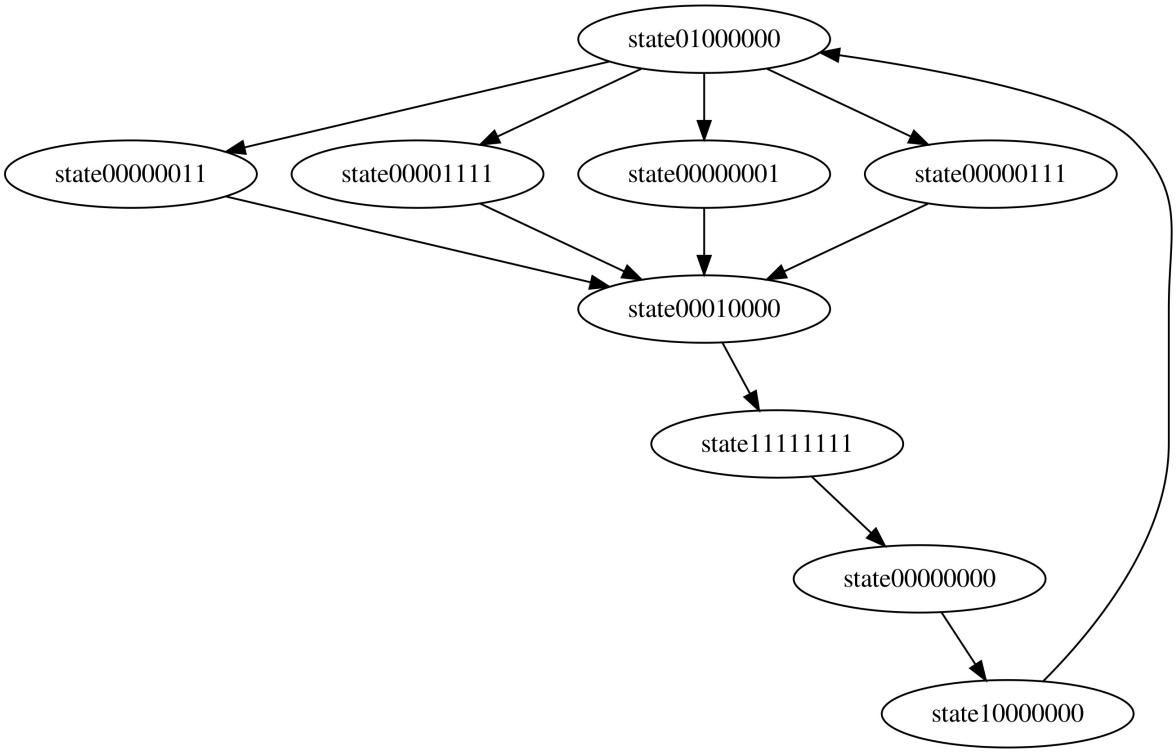
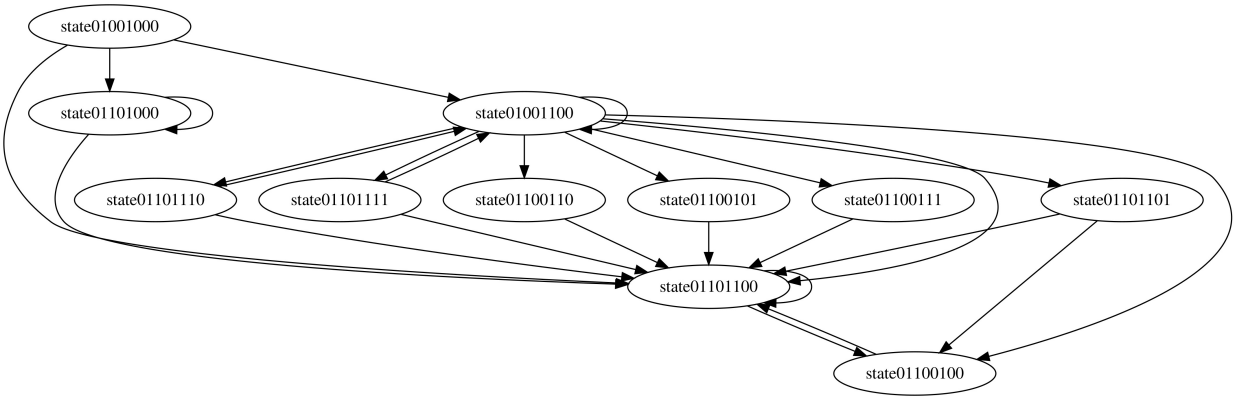


Fig. 4.2 Obfuscated FSM of the warmup benchmark



Next State Register	Current State Register
0 36268	0 36256
36259	36229
36155	36160
36155	36209
36279	36246
0 36303	0
	36586
36247	36209

Table 4.4 Results for identified current state next state pairs for obfuscated warmup benchmark

Next State Register	Current State Register
0 436437	436392
0 436244	436207
0 436284	436253
0 439790	439722
0 441261	441184
0 440093	440017
0 440299	440213
0 436025	436019

Table 4.5 Results for identified current state next state pairs for obfuscated low benchmark

These results were presented in a global logic locking competition, *Attacks on Logic Locking, Obfuscation, Fine-grain Hardware Redaction, & Routing Table Configuration Capture The Flag 2022* [26] and our team *IITG Hackover* from Indian Institute of Technology, Guwahati secured 2^{nd} position [28].

Using Methodology 2 in Section 3.3.1

The multiplexers whose output is directly going to any flip-flop is taken into consideration and its input, output and select line list is being made and is shown in Table 4.6.

Based on the output of the multiplexers the possible current states for the warmup circuit is being identified and is shown in Table 4.7. Here, we can identify only 4 current state registers whereas there should be 8 current state registers according to the original circuit of the warmup benchmark.

Multiplexer Name	Input 1	Input 2	Select Line	Output
U547	n580	n581	n582	_____1828
U694	__09__2013	n746	n747	__99__1850
U765	n806	n807	n747	__9____1926
U769	n809	n539	n582	__9____1920

Table 4.6 Details of the multiplexers identified in the warmup circuit

Multiplexer Output	Input to register (Current State)
_____1828	_____36250
__99__1850	_____36229
__9____1926	_____36155
__9____1920	_____36160

Table 4.7 Possible current state registers for warmup circuit

Using the select lines *n582* and *n747*, we can identify the flip-flops belonging to the OFSM logic of the obfuscated circuit and is listed in Table 4.8.

Name of register
____0_____0_
____0_____36274
____0_____36271
____0_____36260

Table 4.8 Possible registers in OFSM logic of obfuscated warmup circuit

Chapter 5

Conclusion and Future Work

5.1 Combinaitonal Logic Locking

This work tests and validates the effectiveness of the proposed approach TAOBreak, which is an updated version of the approach in [2]. The execution time of proposed approach is better than the existing baseline SMT attack [2]. The attacks by TAOBreak upon TAO-generated obfuscated circuits and also on locked C codes are successful and that too with less execution time. Therefore, it can be concluded that, the approach given in TAOBreak is effective as well as efficient in unlocking the circuits obfuscated using the TAO [1].

For future scope, the key-dependence scenario observed while analysis of the approach on different test cases will have two view points. From the defence point of view, key-dependence in a locked circuit will protect the circuit from this kind of SAT based attacks. Also, from attack point of view, the approach can be improved further to handle such cases where multiple correct keys are possible for a circuit and return any one of the correct key at the end.

5.2 Sequential Logic Locking

The Dishonest Oracle [3] can be attacked without using the scan chain. The statistical approach is effective and scalable in attacking such circuits by considering it completely, as scan chain access is not present.

A improved methodology needs to be formed for determining the latency of the sequential circuits such that proper input output co-relation can be obtained.

In case of Harpoon [4] [5], a proper structural analysis can reveal the original circuit from the obfuscated circiut.

The implementation for identification of the correct reset state has to be done, otherwise input-output co-relation cannot be obtained. ALso, the methology 2 3.3.1 has to be implemented for getting further results.

References

[1] C. M. Pilato, F. Regazzoni, R. Karri, and S. Garg, "Tao: Techniques for algorithm-level obfuscation during high-level synthesis," *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2018.

[2] C. Karfa, R. Chouksey, C. M. Pilato, S. Garg, and R. Karri, "Is register transfer level locking secure?" *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 550–555, 2020.

[3] N. Limaye, E. Kalligeros, N. Karousos, I. G. Karybali, and O. Sinanoglu, "Thwarting all logic locking attacks: Dishonest oracle with truly random logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1740–1753, 2021.

[4] R. S. Chakraborty and S. Bhunia, "Harpoon: An obfuscation-based soc design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[5] M. M. Rahman and S. Bhunia, "Practical implementation of robust state space obfuscation for hardware ip protection," 2022.

[6] S. Dupuis and M.-L. Flottes, "Logic locking: A survey of proposed methods and evaluation metrics," 2019, pp. 1–19.

[7] "Attacks on hardware logic locking obfuscation capture the flag 2021," *https://helloctf.org/21/*.

[8] "Logic encryption: A fault analysis perspective," in *Proceedings - Design, Automation and Test in Europe Conference and Exhibition, DATE 2012*, ser. Proceedings -Design, Automation and Test in Europe, DATE. Institute of Electrical and Electronics Engineers Inc., 2012, pp. 953–958, 15th Design, Automation and Test in Europe Conference and Exhibition, DATE 2012 ; Conference date: 12-03-2012 Through 16-03-2012.

[9] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing ic piracy using reconfigurable logic barriers," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[10] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference, DAC '12*, ser. Proceedings - Design Automation Conference. Institute of Electrical and Electronics Engineers Inc., 2012, pp. 83–89, 49th Annual Design Automation Conference, DAC '12 ; Conference date: 03-06-2012 Through 07-06-2012.

[11] J. Rajendran, H. Zhang, C. Zhang, G. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, Feb. 2015, publisher Copyright: © 2013 IEEE.

[12] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1069–1074. [Online]. Available: https://doi.org/10.1145/1403375.1403631

[13] R. Torrance and D. James, "The state-of-the-art in ic reverse engineering," in *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '09. Berlin, Heidelberg: Springer-Verlag, 2009, p. 363–381. [Online]. Available: https://doi.org/10.1007/978-3-642-04138-9_26

[14] K. Shamsi, M. Li, D. Pan, and Y. Jin, "Kc2: Key-condition crunching for fast sequential circuit deobfuscation," 03 2019, pp. 534–539.

[15] M. El Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," 11 2017, pp. 33–40.

[16] A. Kaur, S. Saha, C. Karfa, and D. Mukhopadhyay, "Corruption exposes you: Statistical key recovery from compound logic locking," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022, pp. 1–6.

[17] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 471–476. [Online]. Available: https://doi.org/10.1145/3299874.3319495

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

[19] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[20] L. de Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and J. Rehof, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340.

[21] J. Rajendran, A. Ali, O. Sinanoglu, and R. Karri, "Belling the cad: Toward security-centric electronic system design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1756–1769, Nov. 2015, publisher Copyright: © 2015 IEEE.

[22] M. Abderrahman, J. Patidar, J. Oza, Y. Nigam, T. AbdulKhader, and C. Karfa, "Fastsim: A fast simulation framework for high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[23] "Z3 solver : Z3 - the smt solver," *http://z3.codeplex.com/*.

[24] R. Karmakar, S. Chatopadhyay, and R. Kapur, "Encrypt flip-flop: A novel logic encryption technique for sequential circuits." arXiv, 2018.

[25] "Leak me if you can : Does tvla reveal success rate ?" 2017.

[26] "Attacks on logic locking, obfuscation, fine-grain hardware redaction, routing table configuration capture the flag 2022," *https://helloctf.org/22/*.

[27] "Winners page for 'attacks on hardware logic locking obfuscation capture the flag 2021'," *https://helloctf.org/21/winners/*.

[28] "Winners page for 'attacks on logic locking, obfuscation, fine-grain hardware redaction, routing table configuration capture the flag 2022'," *https://helloctf.org/22/winners/*.