International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 06 June-2026 | Impact Factor: 3.5

Reverse Engineering Protection with Security of Input/output Locking System in Machine Learning

Dinesh Kumar cholkar[1], Jitendra Ahir[2] and Laxmi Singh[3]

[1]PhD. Scholar, Department of Electronics & Communication Engineering, Rabindranath Tagore University (RNTU), Bhopal

[2,3] Professor, Department of Electronics & Communication Engineering, Rabindranath Tagore University (RNTU) , Bhopal

How to Cite this Article:

cholkar, D. K. & Singh, L. (2026). Reverse Engineering Protection with Security of Input/output Locking System in Machine Learning. International Journal of Creative and Open Research in Engineering and Management, 02(6). https://doi.org/10.55041/ijcope.v2i6.306

https://doi.org/10.55041/ijcope.v2i6.306

Abstract

The implementation of safety logic for security systems are one of the most important technologies is logic locking System. In this used a XOR gate for security purpose. In this the lock can only opened when the desired key is entered which is given by the XOR in the system. In this paper also used sensitivity based algorithms. This algorithm applies to the attacks model based on the sensitivity of the inputs. By using this get control, execution, and range overheads as takes after to begin with, the maximum frequency of the plan(high frequency). A special kind of nodes called a modified bit stream is loaded into an CW305 Artix FPGA used for profiling. While this FPGA is working, power usage is measured. These measurements are then used to train a neural network DNN and an MLP is used. Both models are trained using power traces collected from the profiling FPGA. After training, the attack stage begins. Power traces are gathered from the device being tested and sent to the models that have already been trained. These models try to guess the output of the function that is being used.

Index Term: Logic locking, Power, Security, RE, overheads

I Introduction

In later a long time, the security of touchy information has ended up progressively critical, particularly for inserted “gadgets and the Web of Things (IoT)”. It is vital to guarantee security & keenness whereas at the same time assembly equipment confinements. For illustration, smartphones are the obligation of circuit originators to prepare increasingly delicate information, guarantee information keenness and avoid spills. As a result, it was by and large recognized that equipment is an imperative portion of an imperative electronic framework and must be secure. A adjust between equipment impediments and security is imperative. In any case, this will lead to plan blunders that can be utilized by assailants [1]. Hence, usage ought to be carefully arranged, particularly in the event that the aggressor has physical get to the gadget. Aggressors can look at the gadget and use a assortment of physical assaults to attain their targets. Subsequently, B. Different measures have been proposed for particular assault vectors, counting side channel examination, mistake infusion assaults, equipment control, and falsifying. Be that as it may, it has been watched that these measures can be effortlessly developed [3]. Compelling measures against assault sorts can really increment the affectability of the circuit compared to other assaults. For illustration, in [4], blame discovery circuitry was included to counter

blunder assaults, which cleverly expanded the data, coming endeavors to improve security against particular assaults may in reality weaken the by and large security of the framework [5,6]. The assailant needs a hidden arrange list and an opened IC that can be utilized as a dark box. The SAT solver shapes a fulfilled condition for input designs that give diverse yields for distinctive keys. These designs are called unmistakable designs.

II Attacks on Logic Locking

Attacks against LL methods have a models of a threats that are most utilized to protected through designing of reverse [5]. LL could be a proposed measure to outsource IC plans by adding extra key. The input of the key target is controlled by the basic bits driven from the on-chip memory during operation, whereas the other input may be a useful arrange. The plan works properly only in case the proper key is conveyed or the output is inaccurate in other ways. The plan is secured from unauthorized get to, as it were the IP proprietor has the proper key. The organize list is blocked at the conclusion of the plan stage by embedding's a key stand. This blocked network list is assist utilized in different steps within the IC plan stream. Unlock/activate the IP proprietor after chip era and testing by stacking the right key into the chip. The taking after are diverse security dangers:

Reverse Engineering

In reverse engineering process used the XOR gate and utilize and mention the proper key for the security. Number of steps are their which present in the flowcharts.

Satisfiability-based (SAT) attack

Figure 1 represent the flowchart of the Satisfiability-based (SAT) attack. The first steps of this is to build the lock and gives the SAT of the secured netlist and its work as the input. By using the functional unit and by the SAT solver decides the command is proper working are not in each cycle. if its originate and functioning proper then connected to the valuable Integrated Circuits to get proper Integrated circuits output. After getting the output comparing input and output for proper matching of whatever function have performed. if any mistake has found then the SAT equation, makes a difference to refine the key look space. In every cycle function have checked, if no incorrect key found then comes to the return key.

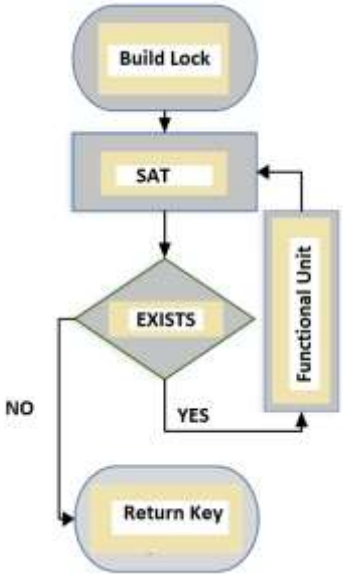

Figure1: Flow chart of the SAT Attack

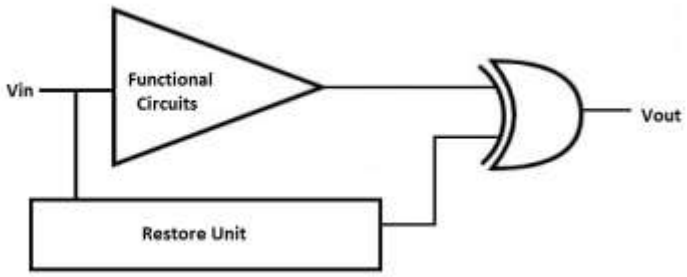

Figure 2: Protection against SAT attack

The proposed figure shown in Fig.3 here shows that how the compare the actual circuits and reverse engineered netlist with a controllable mismatch is produced.In the RE techniques used a "XOR-Based Logic Locking". In this method, key-controlled gates are added to a circuit or design. This ensures that the real function works only with the correct key. To grasp this concept, we look at a logical equation for a locked gate, which is like this.

$Z = (X \oplus K1) \cdot (y \oplus K2)$

Here

'K1, K2' = Design circuits Secret Keys,

' A, B' = Inputs of the logic circuits,

'Z' = Output of the Locked.

By using all these power measurements to get the secret keys, which is part of a simple power analysis attack, and also linking the power data with the input through statistical methods.

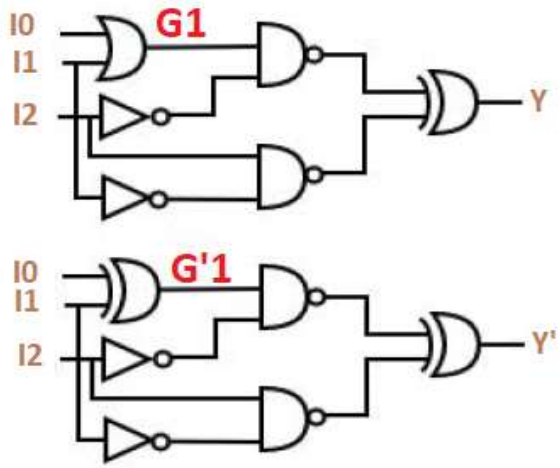

Figure 3: Actual circuit and apply RE by replacing G1.

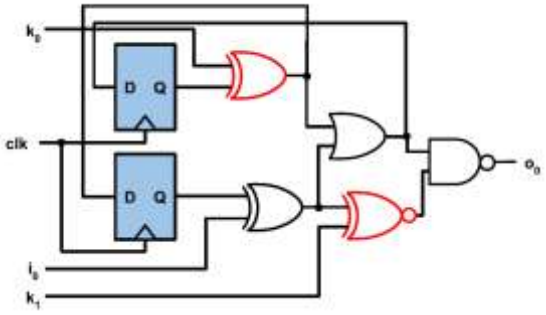

Figure 4: XOR logic locking example

Logic Locking

A guideline thought proposed in this paper is to reuse an existing XOR to right away alter plan inner usefulness once an endeavor to unauthorized get to is identified. Logic locking may be a technique for tending to these fabricating security susceptibilities.

The creators include certain important inputs in a netlist. These inputs are included in the design in a way that works with specific values of those inputs. An attacker would have to guess the correct key from a lot of possible options. Once the chips are built, the correct key is connected to the circuit. Usually, the key is kept in a secure, non-volatile memory. The foundry doesn't know the right key value, so any extra chips made or leaked design data won't work right. One example is XOR logic locking, where XOR gates change signals based on the key input. These XOR gates, which are called key gates, are placed in the design either randomly or through certain methods to make it harder for someone to crack. After the gates are placed, the netlist is redesigned, combining the XOR gates with the existing logic.

Sensitivity optimization

In the field of logic locking research, many different types of attacks have been studied. Because of this, it's very important for anyone using these locking methods to know exactly what kind of attack they are defending against. Other factors to think about include how much computing power the attacker might have and the possibility of using side-channel attacks, like checking electrical signals.

Analysis of Sensitivity based attack algorithms

1. sen :=n;

2.block :=∅;

3. while sen>0 do

Else

Sen := sen-1;

End

4. if SAT[DNN] then

Xproctected := SAT_ASSIGNMENT

end

In this method, the algorithm first identifies inputs that have high sensitivity. Let's say an attack is getting ready with the locked circuit netlist. We look for a part of the circuit that works as the rebuilding unit,

depending on how many inputs. It then needs access to an unlocked IC and the netlist. During each cycle, the key inputs are followed through the circuit. The SAT solver searches for an input that has that "sensitivity level". The "sensitivity" is lowered step by step until a valid input is originate. This valid input is then connected to the prophet, f. If the output matches the result from $f_{flipped}$, a rule is added to a group of blocking clauses called square, and the process keeps going to find the next most sensitive input. If the output doesn't match, it could mean the design is protected.

Evaluation and Performance Analysis

AI-based detection models can be assessed using several measurement methods, such as accuracy, precision, false positive rate (FPR), recall (which is also called true positive rate or TPR), and F1-score, area under the curve (AUC) [1].

"Accuracy" represented by the

$$Accuracy = \frac{TN' + TP'}{TN' + TP' + FP' + FN'}$$

"Precision" represented by the

$$\text{Precision} = \frac{TP'}{FP' + TP'}$$

"True positive rate" represented by the recall

$$\text{Recall} = \frac{TP'}{FN' + TP'}$$

"False Positive Rate" represented by the

$$FPR = \frac{FP'}{TN' + FP'}$$

"F1-Score" represented by the

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

To help DNN and "Machine Learning Detection Models" perform better, these "metrics were applied to multiple datasets from benchmark studies" [2][6].

III Attack Models

In the field of logic locking research, many different types of attacks have been studied. Because of this, it's very important for anyone using these locking methods to know exactly which attack they are trying to prevent. Other factors that matter are how much computing power the attacker has and the possibility of using side-channel attacks, like checking electrical signals.

Our main contributions are:

• For analyzing the "security of logic-encrypted deep networks" with ReLU functions;

• Use an "gate level key bit when the network is contractive".

• To check and fix the "key value for a hidden layer when the network is expansive".

• Used algorithms with DNN for accurate, scalable, and widely applicable it is on real networks.

The SAT solver searches for an input with a specific "sensitivity level". It gradually reduces the sensitivity until it finds a valid input. This valid input is then passed to the predictor, f. "If the output matches the result

from $f_{flipped}$, a constraint is additional to a set of blocking sections called square, and the procedure continues, aiming for the subsequent highest sensitivity input". If the output doesn't match, it might mean the design is protected. For example, if an attack is preparing the locked circuit netlist in advance, we look for a function called flocked in the circuit, which is the rebuilding unit and has several inputs. Throughout all cycles, the key inputs are followed through the circuit. The rebuilding unit is removed from the netlist, creating a new circuit that behaves like $f_{flipped}$. Once we have $f_{flipped}$, we add the "unnormalized sensitivity to the highest value, n, in the circuit".

Implementation of logic locking with reverse engineering

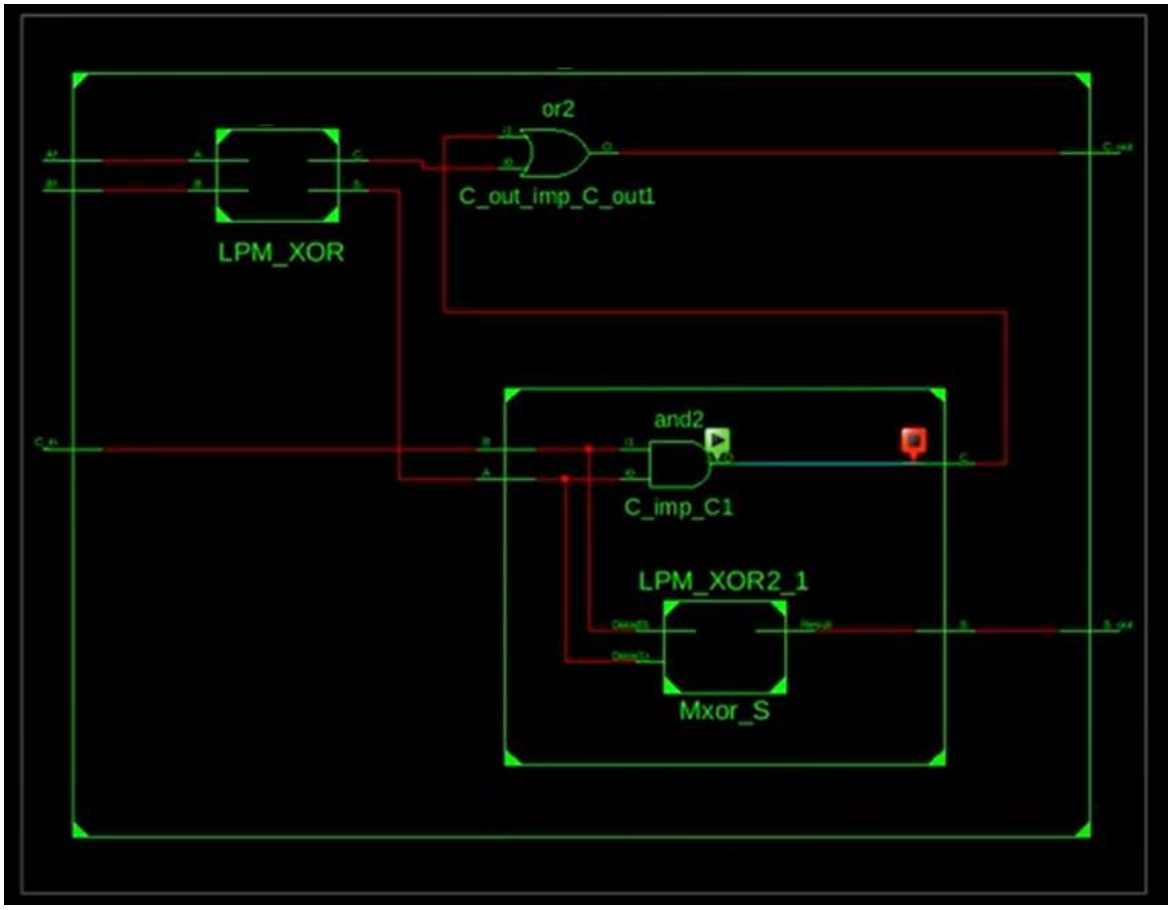

Figure 5 : Implementation of logic locking with reverse engineering

Figure 6 : Timing response of logic locking with reverse engineering

Power Analysis-Based Detection

Techniques look for unusual behavior by examining power usage which they can detect easily.When the system is working normally or if a Trojan is hiding inside, power usage data from the circuit is collected. Then, statistical techniques and machine learning are used to look for patterns that indicate unusual behavior.

$\boldsymbol{P}'(t) = \boldsymbol{I}'(t).\boldsymbol{V}'(t)$

Here

V'(t) = "The instantaneous voltage".

I'(t) = "The instantaneous current".

Power Representation with Machine Learning

The pattern used for the power analysis in the Neural Network represented by the followings.

$$'\boldsymbol{L} = \sum \boldsymbol{I^n} = \boldsymbol{1^n y_i} \,||\, \boldsymbol{f'(x_i) - f'(x_j)} \,||\boldsymbol{2} + \boldsymbol{(1 - y_i) max}\left(\boldsymbol{0, m} - ||\, \boldsymbol{f'(x_i) - f'(x_j)} \,||\right)^{\boldsymbol{2}}$$

Here

xi, xj, = Vectors that represents inputs.

yi = Similarity key label representation.

f'(x) = Extractor representation .

m = Margin representation factor.

This approach is very good at finding power issues that hackers cause.

Security Enhancements

The models used to learn the data at different manufacturing sites without ever sharing sensitive design details.

The rule for combining the models in federated learning is.

$$\theta_T = \sum \boldsymbol{I^n} = \boldsymbol{1^n} \left(\frac{\boldsymbol{m_i}}{\boldsymbol{M}}\right) \boldsymbol{\theta_i}$$

where:

θ_T= The "global model".

m_i = The "fabrication factor of number of samples".

M = The size of the total dataset.

The safe confirms and isolated informs for "AI-based detection", which helps keep improving safety without sharing any data.

Routing Security

To stop interference, electromagnetic leaks, and attacks that use side channels, a special algorithm is used for secure routing, and it is directed by ML.

$$C = \sum (x, y) \in^P w(x, y)$$

Here

P = Possibility of all paths

W(x,y) = The taking into account of security risks with wirelength, cost of routing ,congestion.

The “security-aware routing optimization function” is

min (α’ C’ + β’ S’)

Here

S’ = Susceptibility of side-channel

α', β’ = Performance and security weighting coefficients for balance.

This AI-based routing system helps keep signals safe during transmission and makes it harder for harmful hardware modifications and attacks that try to steal information through side channels [5].

Model of Powered Detection Evaluation

The model uses machine learning to classify things. It was trained using features from the netlist and works by finding the closest examples. This method improved accuracy by 10% compared to older ways that depend only on features.

IV Simulation and Results

The simulation and analysis, result of input/output logic locking for Protection Against Reverse Engineering Attacks using After simulating the attack model and checking the results, assuming that models show certain patterns.

The process started by adding a certain percentage of labels to the node, then splitting it into training and testing parts. After splitting, we trained each model on the training part. Then, we tested the models on the testing part and kept track of performance numbers like “accuracy, precision, recall, and F1 score”. These numbers, along with the percentage, were used to form a new dataset. This new node had the performance numbers as features and the percentage as the label.

This score tells how well the model guesses the level. We made a scoring system for each group of percentages, like [1, 1, 1, 1, 1], like this: a node starts at 0. The results were then used to calculate a score for each level. If the guess is within plus or minus 10% of the real value, the node goes up by 0.5. Then, each set's node is changed by dividing it by the number of guesses, which was 5 in this case. If the guessed percentage is exactly right, the node goes up by 1. If the guess is within plus or minus 5% of the real value, the node goes up by 0.75. The adjusted nodes show how well the model predicts levels, with higher scores meaning better performance. The x-axis shows different types grouped by technique, and the y-axis goes from 0 to 1, showing the scores using our custom system.This was tested and the results are shown in three graphs, covering poisoning ranges of 1-15, 20-35, and 40-50.

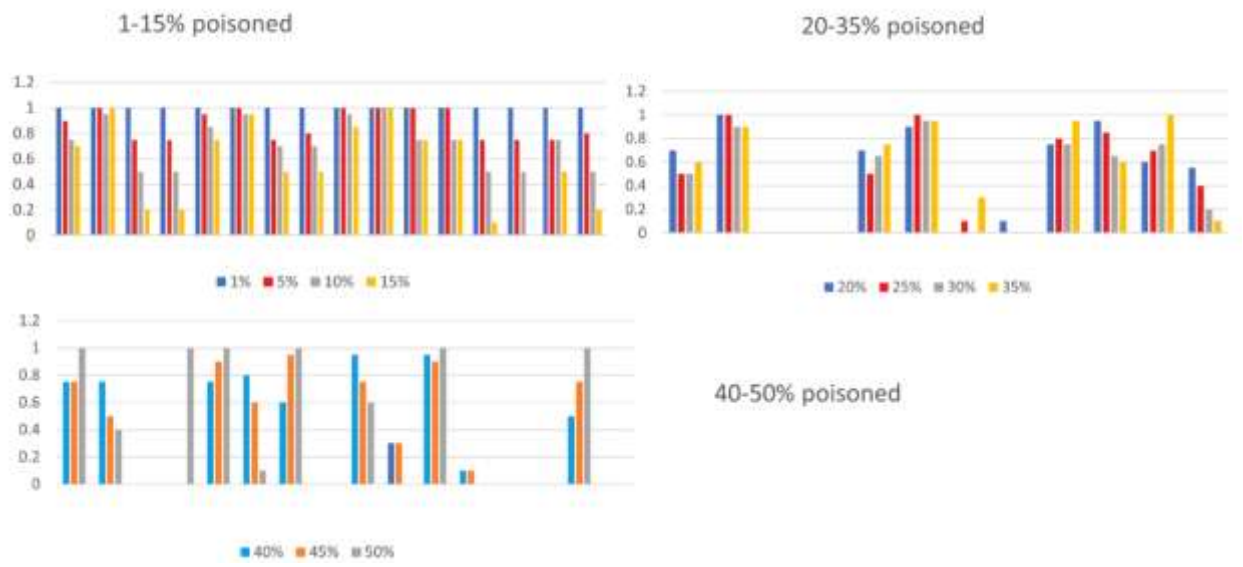

Figure 6: Nodes detection range graph results

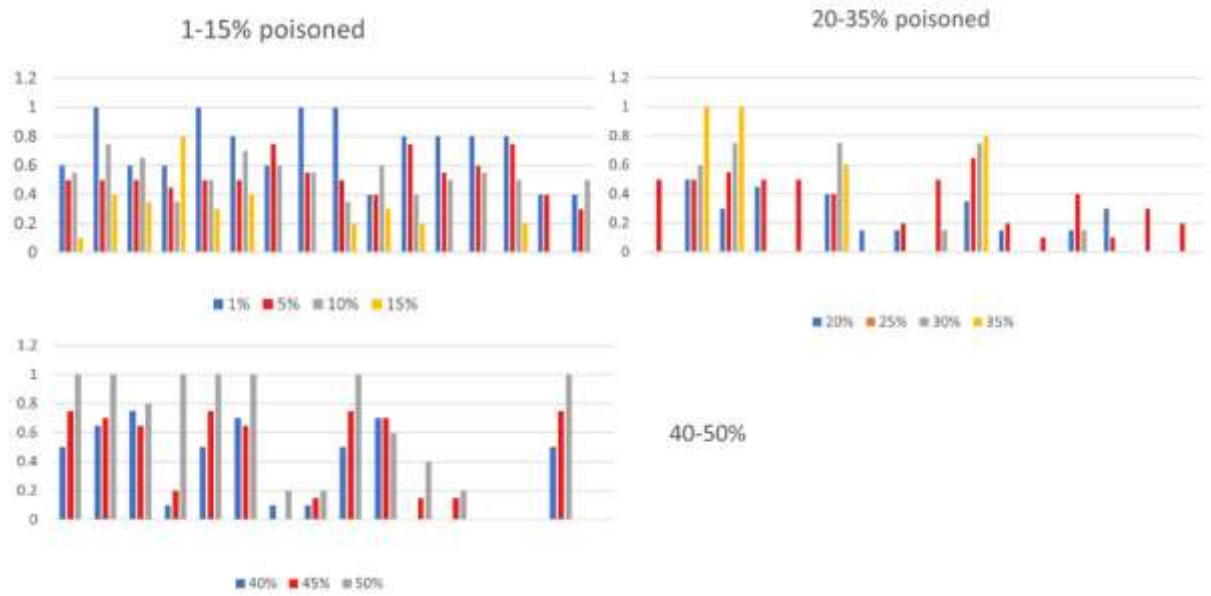

Figure 7 : Detection range nodes MLP Graph results

In figures 6 and 7, the node set scenario shows that the scores stay relatively high when the poisoning rate is low, but they start to decrease as the poisoning rate goes up. When the poisoning rate is very high, the scores drop a lot, which means the system is no longer behaving as expected.

Table:Benchmark SAT evaluation

Benchmark	xor (5-8%) Encryption		dac12 (5-8%) Encryption	
	Decryption time (s)	Decryption iteration	Decryption time (s)	Decryption iteration
c17	n/a	n/a	n/a	n/a
c432	0.057259	3	0.051281	1
c499	0.1015	1	0.115953	5
c880a	0.144556	5	0.157413	25
c1355	0.188653	1	0.393672	2
c1908	3.93094	2	0.673176	29
c2670	0.285035	14	unbreakable	unbreakable
c3540	1.20809	13	3.51718	65
c5315	0.947552	9	9.74864	27
c6288	n/a	n/a	n/a	n/a
c7552	1.51212	10	28.5707	88

Benchmark	xor (5-8%) Encryption		ProbLock (5-8%) Encryption	
	Decryption time (s)	Decryption iteration	Decryption time (s)	Decryption iteration
c17	0.015533	1	0.005433	1
c432	0.05647	3	0.10256	3
c499	0.116477	3	0.2273	9
c880a	0.178235	11	1.6596	43
c1355	0.05869	1	unbreakable	unbreakable
c1908	2.66502	31	2.0384	54
c2670	2.66502	74	145.8	245
c3540	1.51982	18	8.291	82
c5315	4.65455	73	94.024	178
c6288	6.01653	3	1.2284	31
c7552	3.08791	29	674.5	561

After testing ProbLock on several combinational benchmark designs, we found that it offers better security than other logic locking methods.

After looking at and changing the rules for ProbLock, we checked how secure combinational are against a SAT attack. The toc13xor and dac12 benchmarks were encrypted using analysis and methods to find where key gates were added [3]. Based on the fan-in and fan-out ,LCSB is a logic locking test case where key gates are placed of each gate in the logic cone [6]. ProbLock based on XOR is the logic locking method is used.We ran the SAT attack on each combinational benchmark five times and put the average results in Table . The decryption iteration is how many steps the “attack takes to find the key”. The decryption time is how long it takes for the “SAT attack to discovery the correct secret key to open the circuit”. Longer decryption time and more iterations mean the benchmark is more protected from SAT attacks. The most secure benchmarks take the longest time for the SAT attack to finish. If the “SAT attack can't discovery the right key in 30 minutes, the benchmark is measured unbreakable”. As shown in the table, toc13xor, dac12, and LCSB have lower decryption times and fewer iterations compared to ProbLock. ProbLock is much more secure for most circuits. Each encryption method used different ways to keep overhead low, so for every method, the overhead stayed between 5% and 8%.

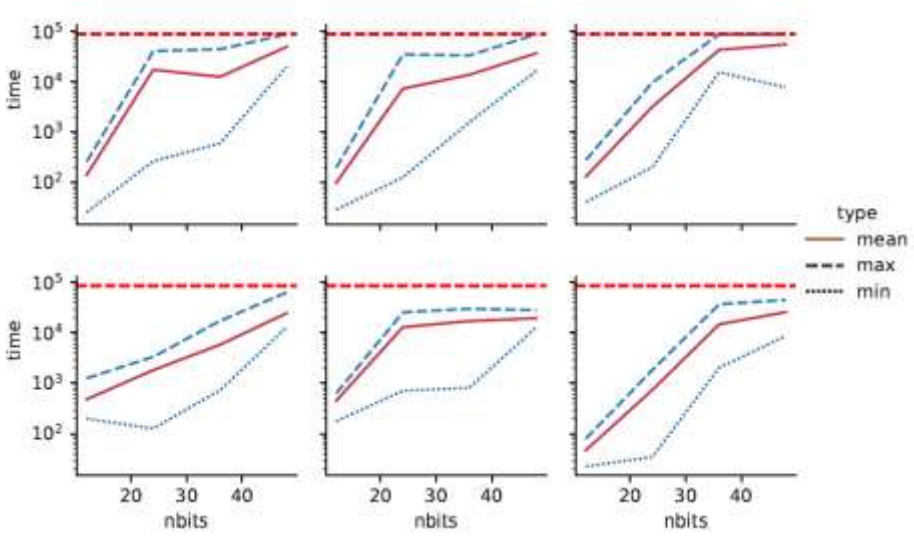

Figure 8: Model checker attack results; timeout of 24 hours indicated by red line.

The results from the model with the attack are shown in Figure 8, which shows how alert the device. "The Power Analysis attack was done by looking at the power uses collected along with the logic outputs to find the expected key values". The highest value key in the logic is taken as the correct one. The right key is the one that is closest to the maximum value, as shown in Figure 8.

Table:Techniques and Technology Used with parameter

Attacks detection mechanisms	Machine learning techniques	Accuracy	Power and area overhead
Reverse Engineering	Deep Neural Network	98%	1%

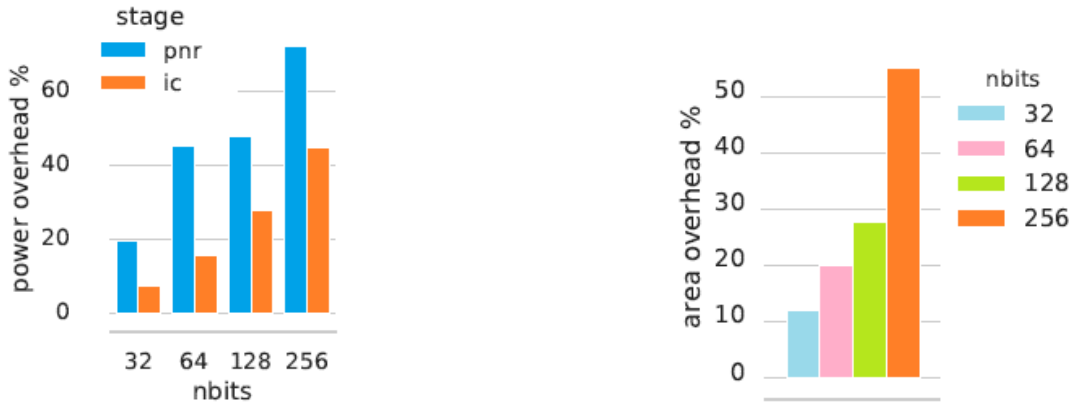

Figure 9 : Power and area overheads

V Conclusion

The FPGA boards used were CW305 were used. Deep learning analysis has two parts. The first part is called the profiling stage, and it happens on a device that the attacker owns. The second part is the attack stage, which happens on the device that is being attacked. In the profiling stage, a special kind of nodes called a modified bit stream is loaded into an FPGA used for profiling. While this FPGA is working, power usage is measured. These measurements are ten used to train a neural network DNN and an MLP is used. Both models are trained using power traces collected from the profiling FPGA. After training, the attack stage begins.The models try

to guess the output of the function that is being used. Power uses are gathered from the device being studied and given to the models that have already been trained.

The results based on different FPGAs and varying numbers of added flip-flops (FFs) due to bit stream changes. Using different FPGAs, especially for training and attacking, makes the results more realistic. One reason is that in a real attack, the attacker probably won't have full control of the target device for the long time needed during the profiling stage. The bigger reason is that training and attacking on the same device causes bias and makes the attack seem more effective than it actually the testing with different numbers of added FFs helps show how the attack's success depends on the number of FFs. The success of an attack on a PUF is measured by how many power traces are needed to make a correct guess.For success is measured by how often the model correctly predicts the output based on just one power trace. The same result each time if the same challenge is used, so multiple power traces can be taken. The fewer traces needed, the better the attack.

References

[1] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in 2013 IEEE Symposium on Security and Privacy. IEEE, 2013, pp. 478–492.

[2] V. Kolesnikov, P. Mohassel, and M. Rosulek, "Flexor: Flexible garbling for xor gates that beats free-xor," in Annual Cryptology Conference. Springer, 2014, pp. 440–457.

[3] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole," in Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2015, pp. 220–250.

[4] K. Shamsi, M. Li, K. Plaks, S. Fazzari, D. Z. Pan, and Y. Jin, "Ip protection and supply chain security through logic obfuscation: A systematic overview," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 24, no. 6, pp. 1–36, 2019.

[5] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 709–720. [Online].Available: http://doi.acm.org/10.1145/2508859.2516656

[6] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu,"Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques," IEEE Transactions on Information Forensics and Security, vol. 12, no. 1, pp. 64–77, 2016.

[7] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in 2013 Design, Automation Test in Europe Conference Exhibition (DATE), March 2013, pp. 1259–1264.

[8] J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra, "A new approach to practical active-secure two-party computation," Cryptology ePrint Archive, Report 2011/091, 2011, https://eprint.iacr.org/2011/091.

[9] S. Kamara, P. Mohassel, and B. Riva, "Salus: a system for server-aided secure function evaluation," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pp. 797–808.

[10] T. P. Jakobsen, J. B. Nielsen, and C. Orlandi, "A framework for outsourcing of secure computation," in Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, 2014, pp. 81–92.