# Work-in-Progress: SAT to SAT-Hard Clause Translator

Rakibul Hassan, Setareh Rafatirad, Houman Homayoun, Sai Manoj Pudukotai Dinakarrao
George Mason University
Fairfax, VA, USA
{rhassa2,srafatir,hhomayou,spudukot}@gmu.edu

## ABSTRACT

Logic obfuscation emerged as an efficient solution to strengthen the security of integrated circuits (ICs) from multiple threats including reverse engineering and intellectual property (IP) theft. Emergence of Boolean Satisfiability (SAT) attacks and its variants have shown to circumvent the security mechanisms such as obfuscation and a plethora of its variants. Considering the size of ICs and the amount of time it takes to validate a defense i.e., obfuscation against SAT attack could range from few ms to days. In contrast, our current work focuses on devising an iterative, dynamic and intelligent SAT-hard clause generator for a given SAT-prone problem. The proposed Machine Learning (ML)-based SAT to unSAT clause translator is a SAT-hard clause generator that utilizes a bipartite propagation based neural network model. The utilized model comprises multiple layers of artificial neural networks to extract the dependencies of literals and variables, followed by long short term memory (LSTM) networks to validate the SAT hardness. The proposed ML-based SAT to unSAT clause translator is trained with conjunctive normal form (CNF) of the IC netlist that are both SAT solvable and SAT-hard. Further, the model is also trained to convert a CNF from satisfiable (SAT) to unsatisfiable (unSAT) form with minor perturbation (which translates to minor overheads) so that the SAT-attack cannot decrypt the keys. To the best of our knowledge, no previous work has been reported on neural network based SAT-hard clause or CNF translator for circuit obfuscation. We evaluate our proposed models's empirical performance against MiniSAT with 300 CNFs.

**ACM Reference Format:**
Rakibul Hassan, Setareh Rafatirad, Houman Homayoun, Sai Manoj Pudukotai Dinakarrao. 2019. Work-in-Progress: SAT to SAT-Hard Clause Translator. In *2019 International Conference on Compliers, Architectures and Synthesis for Embedded Systems Companion (CASES'19 Companion), October 13–18, 2019, New York, NY, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3349569.3351542

## 1 INTRODUCTION

With more semiconductor industries inclining towards fabless business model i.e., outsourcing the fabrication to offshore foundries, to cope-up with the operational and maintenance costs, the threat of hardware security is exacerbating [6]. This hardware threat could be in the form of intellectual property (IP) theft, hardware Trojan (HT) insertions, integrated circuit (IC) tampering, over production and cloning to name a few. Furthermore, the threat could occur during any phase of the IC production cycle ranging from design phase, fabrication phase or even after releasing the design to the market (in the form of side-channel attacks) [6].

To thwart the prevalent security threats, many hardware design-for-trust techniques have been introduced [1] such as split manufacturing [7], IC camouflaging, and logic locking *a.k.a* logic obfuscation. Among multiple aforementioned techniques, logic locking can thwart the majority of the attacks at various phases in the IC Production chain. This is because logic locking requires the correct keys to unlock the true functionality of the design [4].

Although logic locking schemes enhance the security of the IP, the advent of Boolean satisfiability (SAT) based attack [9], also known as "oracle-guided" threat model shows that by applying a few stimuli to the design and analyzing the output, the key value and functionality of an IC could be extracted in the order of a few minutes. To implement SAT attack, the attacker needs access to (a) an obfuscated netlist of IC (obtained after de-layering fabricated IC or constructed from layout), and (b) a functional/activated IC, to which the attacker can apply inputs and monitor the output and functionality. The extracted netlist is converted into a conjunctive normal form (CNF[1]), fed to a SAT solver to determine the keys (assignment to each boolean variable in CNF) to decrypt and reverse engineer the IP.

One of the major challenges with adopting defenses such as obfuscating large number of gates is a large power performance and area overheads [3]. Previous works consider developing Anti-SAT solutions through embedding different metrics (non-convertible to CNF) or through heuristic intuitions and proofs. Such solutions involve challenges including complexity, incompleteness and high probability to exclude parameters that were not explored in literature. To address these concerns, we introduce a neural network with bipartite message passing mechanism that can determine SAT solvability of a given CNF (obtained netlist) and convert the CNF into unSAT (SAT-hard) through minimal modification to the netlist such as flipping a literal (through addition of inverter gate or using XNOR instead of XOR are some of the naïve possibilities) in a clause of CNF.

## 2 SAT TO UNSAT TRANSLATOR

We present the overview of our proposed model in this section. The proposed model is a hybrid Message Passing Neural Network (MPNN) [2] framework that is able to learn SAT deobfuscation by passing messages across the literals and clauses of the given CNF. It learns at a clause-by-clause basis rather than all clauses at once. Furthermore, the proposed model is also equipped with

[1] A CNF is a conjunction (i.e AND) of one or more clauses, where a clause is a disjunction (i.e OR) of literals.


*CASES'19 Companion, October 13–18, 2019, New York, NY, USA*

ACM ISBN 978-1-4503-6925-1/19/10…$15.00
https://doi.org/10.1145/3349569.3351542

a mechanism to modify the netlist i.e., update the CNF in a slight manner to convert the SAT problem to unSAT. Figure 1 depicts the operational flow of the proposed framework.

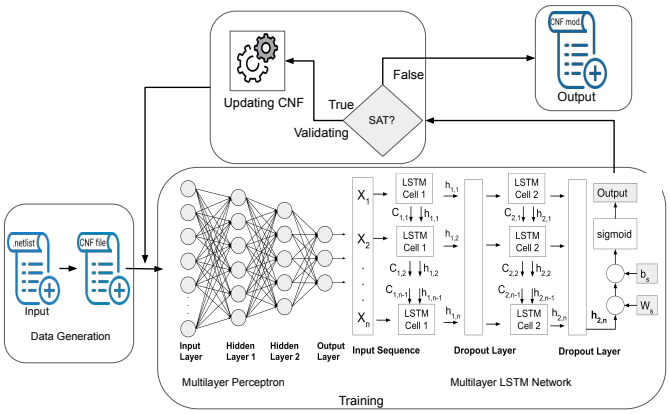


**Figure 1: Architecture of proposed ML-based SAT to unSAT clause translator.**

We provide the CNF form of the netlist for a given IC as an input to the model. We use the MiniSAT [8] SAT-solver as a deobfuscation SAT-attack. The rationale to use this attack is that even the current SAT-attacks are based on the miniSAT phenomenon. However, the model is not limited to nor dependent on MiniSAT attack. As such, one can utilize a different SAT-attack.

The proposed model consists of a literal vector ($L_{init}$) and a clause vector ($C_{init}$) extracted from the CNF, which is fed to a three-layer fully connected MPNN ($L_{msg}, C_{msg}, L_{sat}$), and a two-layer LSTM ($C_u, L_u$) network. Hidden states for literals and clauses are denoted by $L_h$ and $C_h$ respectively. An adjacency Matrix ($M$) defines the relation between literals and clauses. This relationship between literals and clauses are established by connecting edges among them.

Message is passed back and forth along the edges of the network [2]. Multi-layer perceptron (MLP) network takes $L_{init}$ and $C_{init}$ as its input and passes its output ($L_{msg}, C_{msg}$) to the LSTM network [5] which updates the literals $L^{t+1}$ and clauses $C^{t+1}$ at each iteration, as follows:

$$C_u([M^T L_{msg}(L^t)]) \rightarrow C^{t+1} \tag{1}$$

$$C_u([C_h^t]) \rightarrow C_h^{t+1} \tag{2}$$

$$L_u([Flip(L^t), MC_{msg}(C^{t+1})]) \rightarrow L^{t+1} \tag{3}$$

$$L_u([L_h^t]) \rightarrow L_h^{t+1} \tag{4}$$

$L_{sat}$ votes SAT or unSAT for a particular literal and taking the average vote of all the literals after $T$ iteration, SATConda predicts whether a problem is SAT or unSAT.

This message-passing architecture lets the SATConda to learn the features that can distinguish the SAT solvable CNFs from unSAT CNFs. Post training, the proposed SATConda first verifies for the SAT hardness i.e., SAT solvable or unSAT for a given CNF. If the CNF is SAT-hard, it terminates. However, if it is SAT solvable, the SATConda starts perturbing the literals (such as flipping or adding auxiliary variables) beginning from the last clause of a given CNF, thus translating a given CNF from SAT solvable to SAT-hard.

**Table 1: Performance Analysis**

| | |
|---|---|
| Total SAT-problems | 300 |
| SATconda Trained on | 200 problems |
| SATconda Tested on | 100 problems |
| Variables in each CNF | 5 to 20 |
| Trained with | 16 iterations |
| Tested with | 16 iterations |
| Solved problem using the proposed model | 18 |

## 3 EXPERIMENTAL RESULTS

In this section we demonstrate our empirical result of SATConda. Our dataset consists of 300 SAT-problems (200 for training and 100 for testing). The CNFs used in training and test consist of variables ranging from five to twenty. All the considered 100 CNFs in the test scenario are satisfiable (verified through executing MiniSAT solver). Among the 100 tested CNFs through the SATConda, only 18 of the CNFs are solvable through the SAT solver i.e., indicates nearly 82% efficiency in converting SAT to unSAT. In addition to verifying through SATConda, the verification of CNF through traditional SAT solvers is yet to be done. It has been observed that the SATConda perturbs the CNFs slightly so that the CNFs becomes SAT-hard i.e., one can perform the obfuscation efficiently, thus leading to stronger security. Table 1 summarizes the performance of SATConda. More validation of converted CNFs for SAT hardness with other SAT solvers is yet to be performed.

## 4 CONCLUSION AND FUTURE WORK

In this work we introduce a neural network based SAT-hard problem generator towards an intelligent and stronger logic-locking based obfuscation for hardware security. We have observed a successful conversion of SAT to unSAT through the proposed ML-based SAT to unSAT clause translator for multiple variable size CNFs. For future work, we plan to evaluate this framework on state-of-the-art benchmark such as ISCAS benchmark and validate with traditional SAT solvers regarding the SAT hardness. Also, we will improve this model for generating harder SAT-problem so that we can develop existing logic obfuscation more secure from present and emerging attacks.

## REFERENCES

[1] Ferdinand Brasser et al. Special session: Advances and throwbacks in hardware-assisted security. In *Proceedings of the Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, 2018.
[2] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Int. conf. on Machine Learning*, 2017.
[3] Gaurav Kolhe et al. On custom LUT-based obfuscation. In *Great Lakes Symp. on VLSI*, 2019.
[4] Gaurav Kolhe et al. Security and complexity analysis of LUT-based obfuscation from blueprint to reality. In *ACM Int. Conf. on Computer-Aided Design*, 2019.
[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
[6] Marie A Mak. Trusted defense microelectronics: future access and capabilities are uncertain. Technical report, GOVERNMENT ACCOUNTABILITY OFFICE WASHINGTON DC, 2015.
[7] Jeyavijayan JV Rajendran, Ozgur Sinanoglu, and Ramesh Karri. Is split manufacturing secure? In *Conf. on Design, Automation and Test in Europe*, 2013.
[8] Niklas Sorensson and Niklas Een. Minisat v1. 13-a sat solver with conflict-clause minimization. *SAT*, 2005(53):1–2, 2005.
[9] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In *Int. Symp. on Hardware Oriented Security and Trust*, 2015.