Effectiveness of Synthesis/Verification Heuristics in Expediting SAT Attacks

Technical Report
(v1.0, 10/15/2024)

Aric Fowler and Yiorgos Makris (University of Texas – Dallas)

Abstract
The purpose of the work described in this report is to investigate whether the heuristics which are typically employed by logic synthesis and verification packages are able to expedite the well-known SAT attack, which is used to retrieve the key of logic locked circuits. Specifically, this work focuses on the ABC software package and the *fraiging* heuristic which it employs in order to improve the run-time of logic synthesis and verification tasks. While fraiging appears to be effective in logic synthesis and verification, the findings reported herein demonstrate that it is unable to expedite SAT attacks on logic locked circuits.

Problem Definition
We would like to start with a few clarifications:

(i) Toward discovering the functionality of a locked or redacted circuit, the well-established SAT-attack solves a sequence of progressively more complex equivalence checking problems (DIP Identification), followed by a satisfiability problem (Key Solving).

(ii) Recovering the key of a logic locked or redacted circuit is not a synthesis task, since the functionality of the unlocked circuit is not known. Rather, it is a functionality discovery task.

(iii) Synthesis packages, such as ABC, often include verification capabilities which, in turn, rely on equivalence checking. Such equivalence checking may be performed using a satisfiability solver or may also involve other heuristics which may or may not expedite the process of solving the NP-complete satisfiability problem.

Based on the above, the objective of this study is to assess whether the equivalence checking capabilities of a synthesis package such as ABC can be used for launching a cost-effective SAT-attack.

To answer this question, we investigated the capabilities of ABC[1] and discovered the following:

(i) ABC is capable of solving equivalence checking problems for miter circuits, akin to the miter circuits which are constructed for the purpose of DIP Identification during a SAT attack.

(ii) To solve these equivalence checking problems, ABC offers two options: (a) Use a satisfiability solver, which in ABC's case is the *MiniSAT* tool, or (b) Use a combination of *FRAIGING* (Functionally Reducing an And-Inverter-Graph), also known as SAT-sweeping, with a classic satisfiability solver.

(iii) ABC can also be used for the Key Solving step of a SAT-attack, provided that it is formulated as a miter.

Accordingly, the comparison is reduced to the following question:

Can the combination of FRAIGING with SAT-solving expedite SAT-attack, as compared to just using SAT solving? And, furthermore, how would an ABC-based SAT attack compare to a SAT attack based on a contemporary solver, such as Z3?

[1] http://people.eecs.berkeley.edu/~alanmi/abc/abc.htm

Experimental Setup

To answer this question, we implemented a SAT-attack using the capabilities of the ABC tool-suite, as shown in the Figure 1 below. Using this ABC-based flow, we then launched SAT attacks on several logic-locked combinational benchmark circuits, with and without the FRAIGING option offered by ABC. Furthermore, using the same circuits and the same computational environment, we also launched SAT-attacks through the flow shown in Figure 2 below, which is built around the Z3 solver.

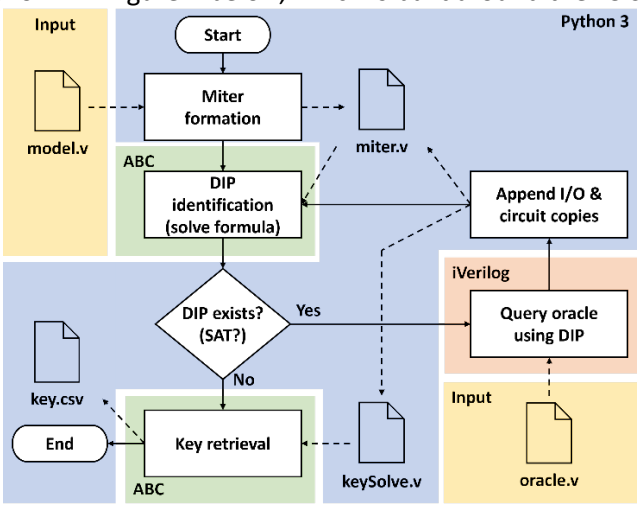

Figure 1: ABC-based SAT Attack Flow

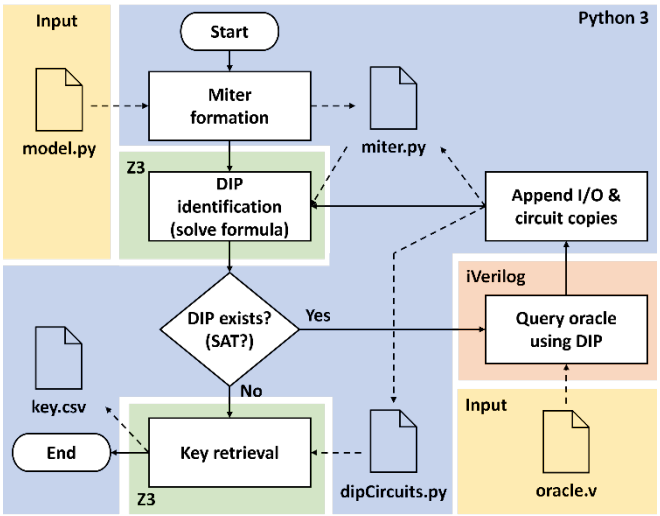

Figure 2: Z3-based SAT Attack Flow

Results

The results are summarized in Table 1, with each entry indicating the fastest among ten runs. Based on the attack run times, it is evident that the combination of FRAIGING and SAT-solving does not offer a time-reduction advantage over just SAT-solving during a SAT attack. Furthermore, since Z3 is a more recent solver that MiniSAT (which has not been updated in over 15 years), it leverages the latest in SAT-solving technology and is therefore significantly (*i.e.* orders of magnitude) faster in the context of a SAT-attack. All

encrypted benchmarks were obtained from Trust-Hub.org[2] and are encrypted with 32-bit secure logic locking[3].

Benchmark Name	ABC (MiniSAT)	ABC (MiniSAT+Fraiging)	Z3 (SMT)
C17	0.553 sec	0.552 sec	0.679 sec
C432	39.1 sec	39.4 sec	20.2 sec
C499	982.3 sec	1003.6 sec	6.0 sec
C880	462.0 sec	472.3 sec	26.8 sec
C1908	408263.7 sec	40820.9 sec	183.2 sec

Table 1: SAT Attack Run-Time Comparison: MiniSAT vs. MiniSAT+Fraiging vs. Z3

Conclusion

Based on the results of this study, we find that the fraiging heuristic employed toward expediting the logic synthesis and verification tasks in the ABC software package does not offer a run-time advantage over using only a satisfiability solver (*i.e.*, MiniSAT). On the other hand, contemporary solvers, such as the SMT-based Z3, offer a significant speedup which can be attributed to the incorporation of the latest technological developments and breakthroughs in these software packages.

[2] S. Amir et al., "Development and Evaluation of Hardware Obfuscation Benchmarks," Journal of Hardware and Systems Security, Jun. 2018.

[3] Yasin, Muhammad, et al. "On improving the security of logic locking." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35.9 (2016): 1411-1424