CAS-Unlock: Unlocking CAS-Lock without Access to a Reverse-Engineered Netlist

Abhrajit Sengupta†, Ozgur Sinanoglu‡
†New York University, ‡New York University Abu Dhabi
{as9397, ozgursin}@nyu.edu

***Abstract*—CAS-Lock (cascaded locking) [1] is a SAT-resilient locking technique, which can simultaneously thwart SAT and bypass attack, while maintaining non-trivial output corruptibility. Despite all of its theoretical guarantees, in this report we expose a serious flaw in its design that can be exploited to break CAS-Lock. Further, this attack neither requires access to a reverse-engineered netlist, nor it requires a working oracle with the correct key loaded onto the chip's memory. We demonstrate that we can activate any CAS-Locked IC without knowing the secret key.**

I. Algorithmic Attack on CAS-Lock

A. CAS-Lock design

Before delving into the attack, let us first analyze the design of CAS-Lock. CAS-Lock works on the same principle of Anti-SAT [2] which is shown in Fig. 1. In Anti-SAT, the outputs of two complementary Boolean functions g and $\overline{g}$ are ANDed together to generate Y, which flips a high observability net in the design such as a primary output (PO). The two blocks g and $\overline{g}$ takes two different n-bit keys k_1 and k_2, respectively, and produces $Y = 1$ for some input patterns for the incorrect keys k_1 and k_2, thereby corrupting the output.

CAS-Lock as shown in Fig. 2, adopts a structure similar to Anti-SAT, where the outputs of two complementary Boolean functions g_{cas} and $\overline{g_{cas}}$ are ANDed together to produce the flip signal Y. The only difference lies in the structure, where CAS-Lock implements a daisy-chained architecture of AND/OR gates as compared to the AND-tree in Anti-SAT. Note that $k_1 = k_2$ produces Y=0, unlocking the chip and ensuring correct (corruption-free) operations.

B. CAS-Lock security

The security of CAS-Lock stems from the difficulty of setting the key values equal, i.e., $k_1 = k_2$, as the bit-wise mapping between the keys k_1 and k_2 is unknown to the attacker. Note that if an attacker is able to figure out this bit-wise mapping between the keys, then setting $k_1 = k_2$ becomes trivial, which leads to a successful attack. This is given by the following equation:

$$\begin{aligned} Y &= g_{cas}(I \oplus k_1) \quad \wedge \quad \overline{g_{cas}(I \oplus k_2)} \\ &= g_{cas}(I \oplus k) \quad \wedge \quad \overline{g_{cas}(I \oplus k)}, \quad k_1 = k_2 = k \\ &= g_{cas}(J) \quad \wedge \quad \overline{g_{cas}(J)}, \quad I \oplus k = J \\ &= 0, \quad \forall I \end{aligned}$$

As can be seen from the derivation above, any attack that can identify the bit-wise mapping between k_1 and k_2 can easily satisfy $k_1 = k_2$ to fix Y to 0 for all input patterns. However, the defense in [1] proposes countermeasures to render the identification of this mapping difficult.

C. Attack on CAS-Lock

In this attack, we demonstrate a scenario where an attacker correctly sets the condition $k_1 = k_2$, without having prior knowledge about the bit-wise mapping between the keys. To this end, she sets both the keys to all zero, which ensures the condition $k_1 = k_2$ for any bit-wise mapping, and bypasses the protection offered by the CAS-Lock block. This is given by:

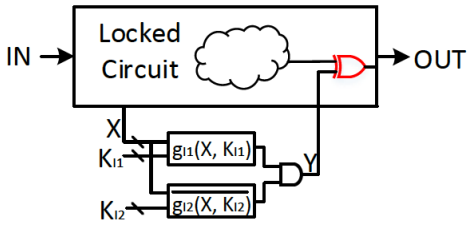

Fig. 1. Architecture of Anti-SAT to generate the flip signal Y. Source: [1].

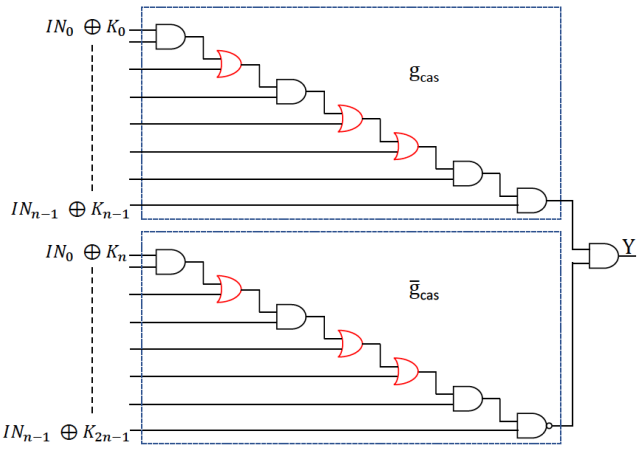

Fig. 2. Architecture of CAS-Lock to generate the flip signal Y. Source: [1].

$$\begin{aligned} Y &= g_{cas}(I \oplus k_1) \quad \wedge \quad \overline{g_{cas}(I \oplus k_2)} \\ &= g_{cas}(I \oplus 0) \quad \wedge \quad \overline{g_{cas}(I \oplus 0)}, \quad k_1 = k_2 = 0 \\ &= g_{cas}(I) \quad \wedge \quad \overline{g_{cas}(I)} \\ &= 0 \end{aligned}$$

Note that she can achieve the same, by setting both the keys to all ones, i.e. $k_1 = k_2 = 0\text{x}F...FF$.

We would like to emphasize that this attack neither requires a reverse-engineered netlist nor a working oracle with the correct key loaded onto the chip's memory. An attacker can simply initialize the key registers with zeros, and the chip becomes immediately functional. Overproduced or counterfeit chips that has CAS-Lock defense can be unlocked using this simple attack.

References

[1] Bicky Shakya, Xiaolin Xu, Mark Tehranipoor, and Domenic Forte. Cas-lock: A security-corruptibility trade-off resilient logic locking scheme. *TCHES*, 2020:175–202.
[2] Yang Xie and Ankur Srivastava. Anti-sat: Mitigating sat attack on logic locking. *TCAD*, 38(2):199–207, 2019.