Open Sesame: A Novel Non-SAT-Attack against CAS-Lock

Akashdeep Saha
IIT Kharagpur
Kharagpur, India
akashdeep@iitkgp.ac.in

Urbi Chatterjee
IIT Kanpur
Kanpur, India
urbic@cse.iitk.ac.in

Debdeep Mukhopadhyay
IIT Kharagpur
Kharagpur, India
debdeep@iitkgp.ac.in

Rajat Subhra Chakraborty
IIT Kharagpur
Kharagpur, India
rschakraborty@cse.iitkgp.ac.in

***Abstract*—CAS-Lock [1], is an advanced logic locking technique that harnesses the concept of single-point function in providing SAT-attack resiliency. It is claimed to be powerful and efficient enough in mitigating state-of-the-art attacks against logic locking techniques. Despite the security robustness of CAS-Lock as claimed by the authors, we expose a serious vulnerability by exploiting the same and device a novel attack algorithm. The proposed attack can reveal the correct key by extracting the *Distinguishing Input Patterns* (DIPs) pertaining to a carefully chosen key simulation of the locked design. The correct key is obtained from the combination of elements from the set of extracted DIPs. Our attack is successful against various AND/OR cascaded-chain configurations of CAS-Lock and reports a 100% success rate in recovering the correct key.**

I. Introduction

The globalization of the integrated circuit (IC) manufacturing industry has resulted in outsourcing of the IC design to foreign fabrication laboratories to reduce time and cost. This has promoted the adversary to come up with various malicious activities in the IC supply chain. Logic locking has risen to prominence as the most efficient defense strategy against such threats throughout the IC supply chain. The basic idea of logic locking is to insert key-gates into the original design to obscure the same to an adversary. The circuit works as intended only on the application of the correct key/s into the locked design.

The advent of Boolean satisfiability-based (SAT) attack [2] marked the beginning of an era of logic locking since SAT-attack could mitigate all the logic locking techniques existing till that time. It is based on pruning the wrong keyspace, such that the existing keys left behind are the correct keys. To mount a SAT-based attack, an adversary requires (i) an oracle or a functional chip and (ii) a locked netlist. The attack proceeds by finding out distinguishing input patterns (DIP), which in turn can eliminate incorrect keys. The DIPs are input patterns that produce different output, on the application of different key values, whereas the oracle helps to distinguish the incorrect keys. There were various techniques proposed to mitigate SAT-attack. However, they were susceptible to other forms of attacks. Recently, CAS-Lock was proposed which showed some promise in resisting most of the attacks against logic locking.

A. CAS-Lock

CAS-Lock [1] is a recently proposed locking technique that differs from Anti-SAT [3] only on the construction of the complementary blocks. Anti-SAT consists of two logic blocks of AND gates arranged in a tree structure, terminated by an AND gate and a NAND gate respectively. The blocks are complementary as for a correct key the two blocks produce complementary outputs. Here, instead of having the AND-tree structure, the AND/OR gates are daisy-chained or cascaded together to form the complementary blocks g_{cas} and $\overline{g}_{cas}$, as illustrated in Fig. 1. Like Anti-SAT, a random combination of XOR/XNOR key gates with the input layer are present in both the blocks of CAS-Lock. Let $K1$ and $K2$ be the keys for g_{cas} and $\overline{g}_{cas}$,

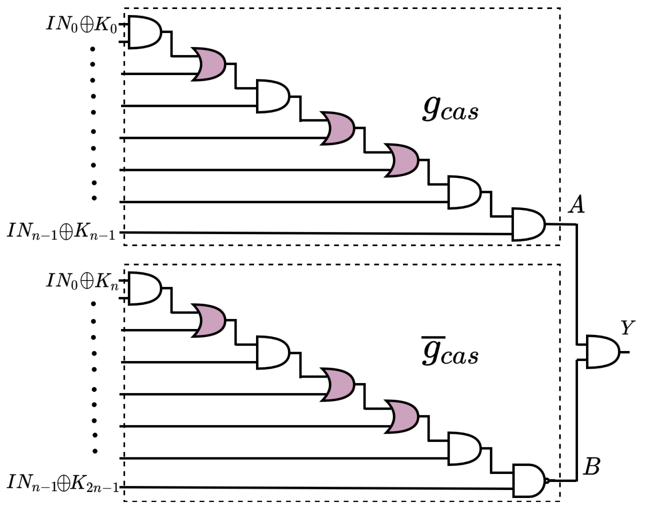


Fig. 1: An instance of CAS-Lock. The blocks g_{cas} and $\overline{g}_{cas}$ are complimentary in nature. The both receive same set of input bits. However, the choice of XOR/XNOR gates are independent in the blocks.

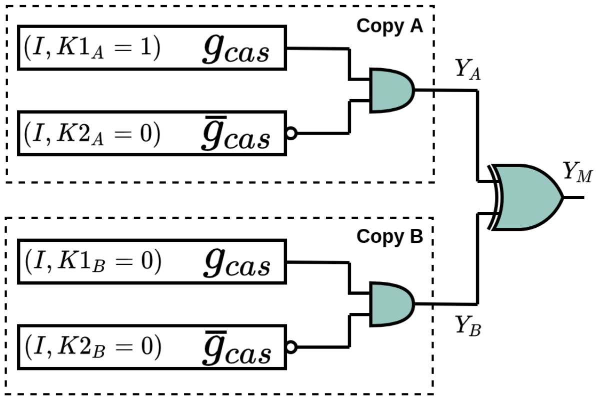
respectively. It offers SAT-attack resistance as it follows the similar SAT-resistance structure of Anti-SAT.

The advantage of CAS-Lock over Anti-SAT is that it could resist the bypass attack [4]. Since Anti-SAT produces a single DIP for an incorrect key, that particular DIP could be easily 'bypassed' utilizing an additional bypass circuitry to produce a working IC. The overhead of the additional bypass circuitry depends on the number of DIPs to bypass. CAS-Lock produces multiple DIPs and hence the overhead of the bypass circuitry becomes infeasible for CAS-Lock. Further, a cascaded version of CAS-Locked (namely M-CAS) could mitigate the removal attack [5].

B. Existing Attacks on CAS-Lock

Since the advent of CAS-Lock, the community is has taken a keen interest in exposing the security vulnerability of CAS-Lock. The author in [6] had proposed a trivial attack to break CAS-Lock by setting key values to all 1's or 0's. This was an ineffective attack algorithm that is based on a misinterpretation of CAS-Lock's implementation. The authors assumed that all the key bits were XORed with the CAS-Lock block inputs. However, they failed to realize that these key-bits were randomly chosen XOR/XNOR gates ($K1 \neq K2$). This insignificant attack was nullified by the authors of CAS-Lock in [7].

Recently, in [8] two variants of attacks on CAS-Lock was proposed to structurally analyze the CAS-Locked netlist and nullify the same. It attempts to identify the flip signal of CAS-Lock locking and eliminates the same by fixing this flip signal to 0/1. However, the adversarial model considered in this work is *very strong* as it assumes that the adversary has full knowledge of the technology mapping and



Copy A
$(I, K1_A = 1)$ g_{cas}
$(I, K2_A = 0)$ $\overline{g}_{cas}$
Y_A
Y_M
Copy B
$(I, K1_B = 0)$ g_{cas}
$(I, K2_B = 0)$ $\overline{g}_{cas}$
Y_B

Fig. 2: Miter circuit used for extracting the DIP set. Copy A and Copy B are the two instances of the locked netlist with different key assignments.

synthesis tools used during locking as specified in Section 2.1 of [8]. Thus, just by concealing the certain synthesis information, the attack complexity could be intensified.

II. Proposed Novel Attack on CAS-Lock

As introduced previously, CAS-Lock follows a structure similar to Anti-SAT but produces more DIPs than Anti-SAT. Let DIP_{set} be the set produced by CAS-Lock corresponding to an incorrect key configuration. We have exploited the relation between the correct key (K_c) of CAS-Lock and DIP_{set}. The proposed attack works towards finding the correct key from the elements of DIP_{set}. *This is complementary to the principles of SAT-attack, where elements of the DIP_{set} set is used to eliminate a set of incorrect keys from the entire probable key space.* Whereas in bypass attacks, the attacker uses to incorporate the bypass circuit for every element of the DIP_{set} set that would restore the correct output of the overall design circuitry.

Further, there exists another exploitable vulnerability in Anti-SAT and CAS-Lock that is: *Pairwise bit-flip of the correct key K_c in the two complementary blocks will also be a correct key*. For example: let $K1 = 0111$ and $K2 = 0101$ be the correct key values. When the first key-bits of the correct key for both the blocks are flipped, $K1^0 = \underline{1}111$ and $K2^0 = \underline{1}101$ be the key-bits in which the 0^{th} key-bit has been flipped with respect to both $K1$ and $K2$. We observe that $K_c = K1^0||K2^0$. Thus, there exists multiple correct key and this observation is valid even when multiple key-bits are flipped.

The attack begins with the DIP_{set} extraction. We follow the same procedure as illustrated in the bypass attack [4] for DIP extraction. It begins with the construction of the miter circuit, as illustrated in Fig. 2 to extract the DIP_{set} for a wisely chosen incorrect key configuration for the complementary blocks. The miter circuit consists of two copies of the locked netlist: Copy A and Copy B. Copy A has $K1_A = 1$ and $K2_A = 0$, whereas Copy B has both $K1_B = 0$ and $K2_B = 0$ as the chosen key values. This choice of key value assignments to the miter circuit prevents generation of overlapping DIPs. We are interested in DIPs generated for Copy A, that is for $K1_A = 1$ and $K2_A = 0$, to get K_c.

Once DIP_{set} is constructed we proceed to identify the correct key from it. DIP_{set} always consists of odd number of elements [1]. There is exactly one element in DIP_{set} which is non-repeating. We identify this DIP and this corresponds to $K1$. We just require to query the oracle with other elements of DIP_{set} to get $K2$. We illustrate our attack on a miniature circuit of stand-alone CAS-Lock blocks.

Example: Let us take into account the CAS-Locked netlist as shown in Fig. 3 and fed into our miter circuit construction (as

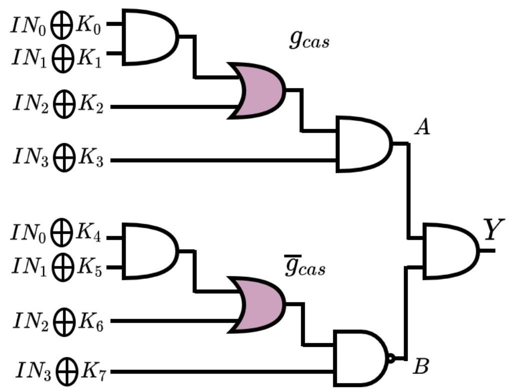


Fig. 3: An instance of CAS-Lock technique for 4 inputs. K_0, K_1, K_2, K_3 are collectively represented as $K1$ is the key input to the g_{cas} block. Likewise, K_4, K_5, K_6, K_7 are represented as $K2$ is the key input to the $\overline{g}_{cas}$ block. Both the blocks share same set of inputs: IN_0, IN_1, IN_2, IN_3. A and B are single-bit outputs of g_{cas} and $\overline{g}_{cas}$ blocks, respectively. and Y is the final output of the CAS-Lock block. The key-bits K_1, K_2, K_3, K_5, K_7 are XNORed with the inputs. Key-bits K_0, K_4, K_6 are XORed.

referred in Fig. 2). The $DIP_{set} = \{0011,\ 1011,\ 0101,\ 0111,\ 1111\}$ corresponding to Copy A. It is evident that $X011$, $X111$, and 0101 have been obtained as DIPs, where X signifies don't care at MSB. It is seen that $K_c = 01010111$, where $K1 = 0101$ and $K2 = 0111$ exists in DIP_{set} and 0101 is the non-repeating DIP. Further, $K_c = 01110101$ is also a correct key where the key values of $K1$ and $K2$ are exchanged. The proposed attack is also effective for OR/NOR CAS-Lock structures.

III. Conclusion

In this work, we have demonstrated a novel attack algorithm that is effective yet simple in unlocking CAS-Lock. The assumptions and the interpretation of the security analysis of CAS-Lock made in this work conforms with the original work. This attack is potent enough to extract the secret key for various CAS-Lock configurations simply by analyzing the DIPs for an incorrect key. It is devoid of any requirement of complex and often infeasible structural analysis of the locked netlist.

References

[1] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "Cas-lock: A security-corruptibility trade-off resilient logic locking scheme," *IACR Trans. CHES*, pp. 175–202, 2020.

[2] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *HOST*. IEEE, 2015, pp. 137–143.

[3] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Trans. CAD*, vol. 38, no. 2, pp. 199–207, 2018.

[4] X. Xu and et. al., "Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks," in *CHES*. Springer, 2017, pp. 189–210.

[5] M. Yasin and et. al., "Removal attacks on logic locking and camouflaging techniques," *IEEE Trans. Emerging Topics in Computing*, vol. 99, no. 8, pp. 1–14, 2017.

[6] A. Sengupta and O. Sinanoglu, "Cas-unlock: Unlocking cas-lock without access to a reverse-engineered netlist." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1443, 2019.

[7] B. Shakya, X. Xu, M. M. Tehranipoor, and D. Forte, "Defeating cas-unlock." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 324, 2020.

[8] A. Sengupta, N. Limaye, and O. Sinanoglu, "Breaking cas-lock and its variants by exploiting structural traces."