SigAttack: New High-level SAT-based Attack on Logic Encryptions

Yuanqi Shen, You Li, Shuyu Kong, Amin Rezaei, and Hai Zhou
Northwestern University
{yuanqishen2020, you.li, shuyukong2020}@u.northwestern.edu, me@aminrezaei.com, haizhou@northwestern.edu

***Abstract*—Logic encryption is a powerful hardware protection technique that uses extra key inputs to lock a circuit from piracy or unauthorized use. The recent discovery of the SAT-based attack with Distinguishing Input Pattern (DIP) generation has rendered all traditional logic encryptions vulnerable, and thus the creation of new encryption methods. However, a critical question for any new encryption method is whether security against the DIP-generation attack means security against all other attacks. In this paper, a new high-level SAT-based attack called SigAttack has been discovered and thoroughly investigated. It is based on extracting a key-revealing signature in the encryption. A majority of all known SAT-resilient encryptions are shown to be vulnerable to SigAttack. By formulating the condition under which SigAttack is effective, the paper also provides guidance for the future logic encryption design.**

I. Introduction

Globalization of the integrated circuit (IC) design industry leads to severe issues in the field of hardware security such as overproduction, piracy, reverse engineering and counterfeiting [5]. As a countermeasure, logic encryption is proposed to add extra key inputs into the design such that even though attackers can know the netlist, the circuit is functional only when the key inputs are set correctly [1]. Diverse logic encryption techniques have been proposed [1]–[4], [6], [14]. However, they are all vulnerable to a newly discovered SAT-based attack [13], and based on such discovery, many new encryption and decryption techniques are proposed [7]–[12], [15]–[17]. It naturally draws us to this critical question: is the SAT-based attack with DIP (Distinguishing Input Pattern) generation [13] the most powerful attack, and is its measure of attack complexity reliable?

Our answer is no. In this paper, we first define a key-revealing signature in any logic encryption, and show that if attackers know such a signature, they can easily find the correct key. Then we introduce SigAttack, a new high-level SAT-based attack, and show how SigAttack extracts such signatures from some well known encryption schemes, especially the designs proposed by Zhou [18]. The general condition under which such a signature can be extracted will also be discussed thus providing advice for future logic encryption designers.

II. Related Works

Zhou [18] has proposed a theory on logic encryption, which provides a deep understanding on the design space and the trade-off between error rate and attack complexity. It can be proved that in any given encryption $C(X, K, Y)$ for any function $F(X)$, if the minimal error number of a wrong key is M, the minimal attack complexity is N, then MN $\leq 2^n$, where n is the length of inputs [18]. Therefore, one ideal encryption shown in Figure 1 is to set both the minimal attack complexity and the minimal error number for a wrong key to $2^{n/2}$, and

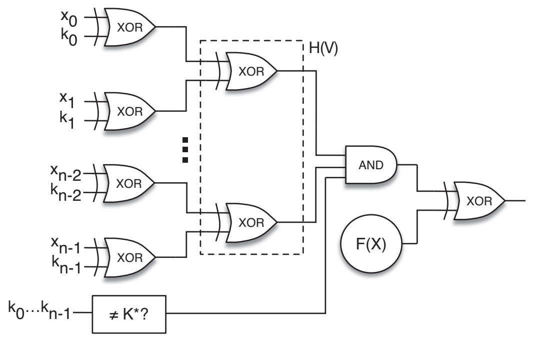

Fig. 1. A logic encryption design with both high error number for a wrong key and high attack complexity.

Zhou has proved that such a design $C(X, K, Y)$ exists for any given $F(X)$.

To generalize the design, Figure 2 is proposed, and Zhou [18] has shown that every logic encryption is functionally equivalent to this general scheme. In this design, X and K are primary inputs and key inputs, respectively, and K^* is the correct key value. If K is equal to K^*, the flipping signal is disabled. Otherwise, the value of the flipping signal only depends on the output of the function $H(X, K)$, and it may output one under different combination of X and K. It can be checked that all encryption techniques conform to this scheme. For example, $H(X, K)$ is combination of XOR gates and a NOR gate in SARLock, as shown in Figure 3.

III. SigAttack: High-level SAT-based Attack

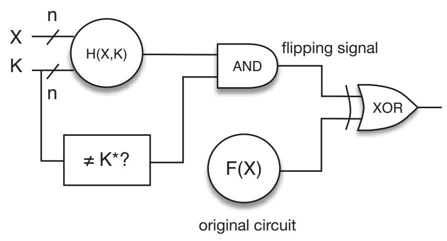

Fig. 2. The general scheme that every possible logic encryption is equivalent to.

A. Signature Definition

To develop a logic decryption technique against the generalized model in Figure 2, we first conduct the structural analysis. We follow the same assumption in [18] that the outputs of functions $H(X, K)$ and $F(X)$ have only one bit. We noticed that if the output of the function $H(X, K)$ can be

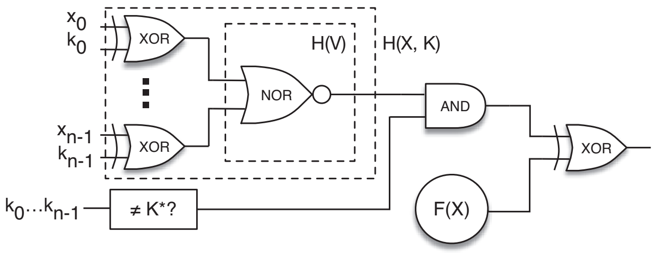

Fig. 3. Model SARLock in the general framework and Zhou's encryption.

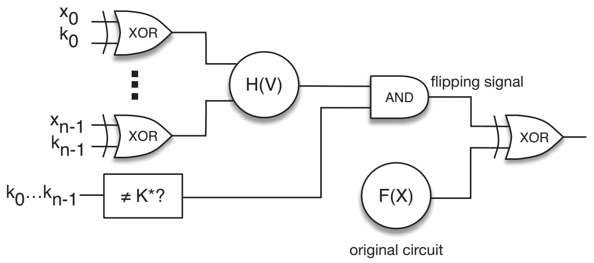

Fig. 4. Zhou's encryption.

fixed to one, the value of the flipping signal only depends on the correctness of the key. Therefore, if we require different outputs of two copies of the encrypted circuit, one of the keys must be equal to K^*. In other words, the correct key can be revealed by a SAT query $C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge (Y_1 \neq Y_2) \wedge H(X, K_1) \wedge H(X, K_2)$. However, designers can further obfuscate the encrypted circuit so that the netlist of $H(X, K)$ is hard to be extracted. Is an attacker able to fix the output of $H(X, K)$ to be one without knowing the netlist of H?

The answer is positive. Intuitively, a signature of an encryption in Figure 2 is a function $Sig(X, K)$ that implies $H(X, K)$. We formally define the key-revealing signature of an encrypted circuit, $Sig(X, K)$, as follows:

Definition III.1. *For any given encryption circuit $C(X, K, Y)$, a signature $Sig(X, K)$ is defined as any Boolean expression such that:*

1) $\forall X, K, K \neq K^* : Sig(X, K) \implies C(X, K, Y) \wedge (Y \neq F(X))$;
2) $\exists X, K, K \neq K^* : Sig(X, K^*) \wedge Sig(X, K)$.

Intuitively, condition 1 ensures that $Sig(X, K)$ captures a subset of (X, K) pairs that will always produce wrong outputs when $K \neq K^*$ in $C(X, K, Y)$. In the general encryption shown in Figure 2, it means that $Sig(X, K)$ must be an implicant of $H(X, K)$. However, any (X, K) combination giving a wrong output can form such an implicant. Therefore, condition 2 requests that $Sig(X, K)$ must contain at least two pairs (X, K^*) and (X, K). It excludes any simple instance (X, K) as $Sig(X, K)$. The vulnerability of a logic encryption with known $Sig(X, K)$ is shown in the following theorem.

Theorem III.1. *If attackers can extract the $Sig(X, K)$ for a given encryption circuit $C(X, K, Y)$, the correct key value K^* can be revealed by a SAT query $C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge Sig(X, K_1) \wedge Sig(X, K_2) \wedge (Y_1 \neq Y_2)$.*

B. SigAttack on Zhou's Encryption

Based on the design in Figure 2, a general encryption scheme in Figure 4 has been proposed by Zhou [18], which achieves linear-size encryption by using XOR gates to pair bits between primary inputs X and key inputs K. We called it Zhou's encryption in this paper. It is easy to check Figure 1 and 3 conform to Zhou's encryption.

Could we attack Zhou's encryption without knowing the specific design pattern of the function $H(V)$? Similar to the discussion in Section III-A, if the output of $H(V)$ equals one, the flipping signal only depends on the correctness of K, thus K^* can be revealed by the SAT solver. Therefore, the question further becomes how we could fix the output of $H(V)$. It turns out we do not need to fix any value of X and K; since each bit of X and K is XORed first, fixing the equivalent relation of each bit between X and K is enough to provide the same input V to $H(V)$. However, we should assume attackers do not know how bits are paired between X and K. In other words, we should explore for each bit x in X, which key bit k in K is connected to the same XOR gate. If we know the connection for all bits of X and K, we can add clause either $(x_i = k_i)$ or $(x_i \neq k_i)$ for each bit of X and K to a SAT solver so that V is determined.

To find the bits pairing, we propose the following SAT-based technique: for each bit k in K, we query the SAT formula

$$C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge (Y_1 \neq Y_2) \wedge \\ (k_1 \neq k_2) \wedge ((K_1 \backslash k_1) = (K_2 \backslash k_2)),$$

which indicates if exists two key values K_1 and K_2 with hamming distance equal to one, such that $H(V) = 1$ for one circuit and $H(V) = 0$ for the other. As a result, their outputs are different since one of them is flipped. Here we assume that both K_1 and K_2 are unlikely to be assigned to the correct key K^* by a SAT solver. Otherwise, K^* is already revealed.

If the output is not flipped because $H(V) = 0$, we know after flipping the corresponding primary input bit x that connects to the same XOR gate with k, $H(V)$ is equal to one again, so the flipping signal is enabled for both circuits. Thus, to find such x, we flip each bit of X and observe if the output Y is flipped again.

What could go wrong here? Since we do not know the function of $H(V)$, we cannot guarantee that Y is flipped only when we flip the corresponding primary input bit x. Flipping other primary input bits can produce a different V, which may also lead to $H(V) = 1$. Thus, we may not find the correct pairing. Inspired by this discovery, we can conclude that Zhou's encryption is vulnerable if it has the following property:

Property 1. $\exists$ *at most one V' such that $H(V') = 1$ if $H(V) = 0$ and $hamming\_distance(V, V') = 1$.*

However, what if such V' does not even exist? In other words, the SAT solver is not able to find an assignment of the proposed query for some key input bits. As a result, we miss pairings for some bits in X and K, and the doubt is if these missing pairings lead to the result that $H(V) = 1$ for two copies is not guaranteed. Fortunately, we can prove the Theorem III.2.

Theorem III.2. *For the key bits that cannot satisfy the proposed SAT query, the value of these key bits cannot affect the output of the function H(V).*

Theorem III.2 guarantees that if the SAT query cannot be satisfied for a key bit, we can simply skip it and check the next bit. If Property 1 holds, after iterating all key bits, we are able to collect all constraints of the relation between X and K to guarantee $H(V) = 1$. Therefore, the SAT solver can be utilized to directly find K^*.

However, the algorithm is still not perfect. There may be a case that the SAT query is satisfied for all key bits, and it leads to the SAT solver returns UNSAT (unsatisfiability) while finding the K^*. The reason is that the SAT solver finds the same primary inputs for both copies of circuits, and if $H(V) = 1$ requires all key bits to be constrained, K_1 and K_2 have to be assigned the same value. As a result, the SAT solver cannot find an assignment to satisfy the clause $(Y_1 \neq Y_2)$ in the SAT query.

To overcome this issue, the SAT solver is asked to find two distinct assignment V_1, V_2 such that $H(V_1) = H(V_2) = 1$, which allows us to add constraints between X and K_1 based on V_1, and X and K_2 based on V_2. Therefore, the SAT solver has the flexibility to find different assignments to K_1 and K_2. If the SAT solver cannot find the second assignment, it simply indicates that there is only one possible V such that $H(V) = 1$, and any wrong key has exponential low error rate. Hence, we could simply consider a random key to be correct and fix a wrong output by bypass attack [16].

The algorithm of SigAttack on Zhou's encryption is as follows. First, the pairing between each bit of X and K is found, and we add constraints to a SAT solver to guarantee that outputs of $H(V)$ for both circuits, $H(V_1) = H(V_2)$, are equal to one. Such V_1 and V_2 with different values can be found by SAT queries and comparing SAT assignments of bits in each found pair. Therefore, the correct key value can be found by one SAT query. If the second assignment V_2 cannot be found, we conduct bypass attack. As a result, we can prove the following theorem:

Theorem III.3. *Zhou's encryption shown in Figure 4 can be attacked by SigAttack if Property 1 holds.*

C. SigAttack on SARLock

The analysis from Section III-B indicates that SigAttack can attack SARLock [17]. We have developed Figure 3, which is equivalent to SARLock. $H(V)$ is simply a NOR gate; if primary input X and key input K have the same value, the flipping signal is enabled as long as K is not equal to the correct key K^*.

Let us run SigAttack on this design. We can pair each bit x_i and k_i for i from 0 to $n-1$. However, we can not find two different inputs V_1 and V_2 such that $H(V_1) = H(V_2) = 1$; the output of NOR gate can be one only if its inputs are all zeros. It is reasonable since there is only one wrong input-output pair for the SARLock design when the key value is incorrect. Since we cannot find the second assignment of V, we immediately know that any wrong key has exponentially low error rate. Therefore, we select a random key and fix a wrong input-output pair by bypass attack [16].

D. SigAttack on Anti-SAT

We model an Anti-SAT design [15] to Zhou's encryption as shown in Figure 5, and $H(V)$ is the original Anti-SAT blocks. The design is similar to Zhou's encryption, but with a little modification: The original Anti-SAT design does not fix a correct key value; instead, the key value is correct as long as $k_0k_1...k_{n-1}$ are equal to $k_nk_{n+1}...k_{2n-1}$. Therefore, there are 2^n correct key values. Meanwhile, the primary input X is compared twice, with $k_0k_1...k_{n-1}$ and $k_nk_{n+1}...k_{2n-1}$.

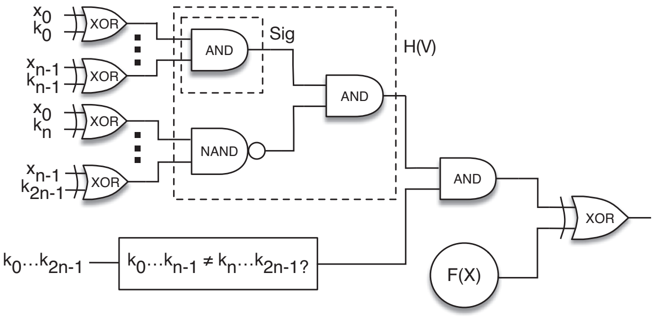

Fig. 5. Model Anti-SAT in Zhou's encryption.

Since Figure 5 is not exactly fit for the model of Zhou's encryption, could we still attack it by SigAttack? The answer is yes. The first step is to find pairings between each bit of primary inputs X and key inputs K. If the key input bit k_i that we flipped is in $k_0k_1...k_{n-1}$, we can only flip the corresponding primary input bit x_i to restore the output of $H(V)$ from zero to one. Therefore, we are able to find pairings between x_i and k_i when $0 \leq i \leq n-1$.

However, when $n \leq i \leq 2n-1$, flipping any x_i can restore the output of $H(V)$ from zero to one, since the output of NAND gate is zero if and only if its inputs are all one. As a result, we may not find the correct pairing between x_i and k_i when $n \leq i \leq 2n-1$. Fortunately, SigAttack can find the correct key without knowing these pairings. After we add constraints between x_i and k_i for $0 \leq i \leq n-1$, we can directly find a correct key by a SAT query. It is because if the output of the NAND gate is zero, $x_i \neq k_{n+i}$ for all $0 \leq i \leq n-1$. Since the output of the AND gate (indicated as Sig in Figure 5) is fixed to be one, $x_i \neq k_i$ for all $0 \leq i \leq n-1$. Thus, $k_0k_1...k_{n-1} = k_nk_{n+1}...k_{2n-1}$, which indicates the key is correct. Since $Y_1 \neq Y_2$, either K_1 or K_2 is the correct key, which can be easily identified by comparing Y_1 and Y_2 with the correct output.

IV. Experimental Results

We evaluate SigAttack on Zhou's encryption, SARLock and Anti-SAT. Our experiment is conducted on a machine with Intel core i5 clocked at 2.4 GHz and memory 5.8 GB. The original benchmarks are from the ISCAS'85 and the Microelectronics Center of North Carolina. We build benchmarks of Zhou's encryption shown in Figure 1, and various input lengths are tested to comprehensively show the effectiveness of SigAttack. We also build Zhou's version of SARLock and Anti-SAT shown in Figure 3 and Figure 5, respectively. For comparison, we choose the SAT-based attack [13] and Double DIP [12] as the representative of exact and approximate attacks, and compare the correctness and accuracy.

We perform SigAttack, the SAT-based attack, and Double DIP on Zhou's encryption. *The experimental result indicates that the correct key K^* of all benchmarks can be successfully decrypted by SigAttack within a few seconds.* Since Double DIP is an approximate attack, out of 9 benchmarks that Double DIP can finish within our time limit (5 hours), only 5 benchmarks are decrypted with the correct key. Figure 6 demonstrates the relation between the execution time and

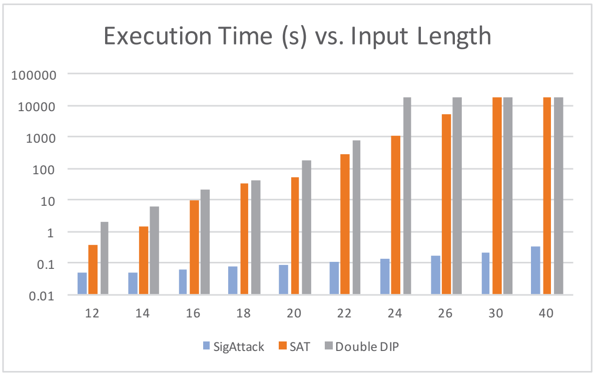

Fig. 6. Execution time of performing SigAttack on Zhou's Encryption compared with the SAT-based attack and Double DIP.

the input length. We can see that SigAttack takes much less execution time while keeping the accuracy compared with the SAT-based attack and Double DIP. With the increasing of the input length, the execution time of the SAT-based attack and Double DIP dramatically increases, and most of benchmarks cannot be decrypted within 5 hours.

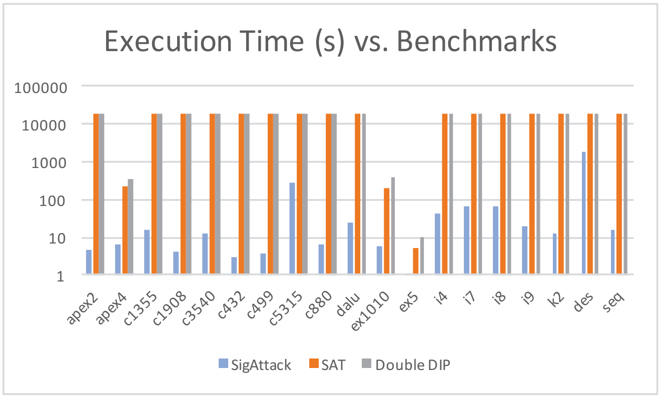

Fig. 7. Execution time of performing SigAttack on Anti-SAT compared with the SAT-based attack and Double DIP.

We perform sigAttack on SARLock and Anti-SAT shown in Figure 3 and Figure 5. As we analyzed in Section III-C, for SARLock, SigAttack reports it cannot find two different inputs V_1 and V_2 to $H(V)$ such that $H(V_1) = H(V_2) = 1$. Therefore, bypass attack is conducted, and [16] already shows that bypass attack can efficiently decrypt SARLock. However, the SAT-based attack cannot finish most of benchmarks within our time limit (5 hours), and even though Double DIP can finish running quickly, the solved keys for all of benchmarks are incorrect. *On the other hand, Figure 7 indicates SigAttack can successfully decrypt all benchmarks encrypted with Anti-SAT within at most a few minutes.* In contrast, SAT-based attack and Double DIP cannot decrypt most of benchmarks within 5 hours. For benchmarks apex4, ex1010 and ex5 that Double DIP can finish on time, correct keys are found.

V. Discussion

The development of SigAttack raises such a question: to decrypt an encryption, do we need to know its exact design pattern? As we have shown, SigAttack does not depend on knowledge of functions $H(X,K)$ in Figure 2 and $H(V)$ in Figure 4. Therefore, from the perspective of attackers, instead of exploring the exact design pattern, it is more important to extract the signature $Sig(X,K)$ of an encryption circuit $C(X,K,Y)$. Section III provides examples of extracting signatures of different encryption techniques.

Another discovery is that we may reconsider the robustness and reliability of XOR encryption in digital designs. From Section III-B to Section III-D, XOR operations between primary inputs X and key inputs K become a vulnerability explored by SigAttack, since it provides the flexibility for a SAT solver to assign specific values while maintaining desired outputs. Designers should carefully analyze if their encryption can be modeled to the vulnerable model, Zhou's encryption, by resynthesis. If yes, Property 1 should be avoided.

VI. Conclusion

In this paper, we have developed a new high-level SAT-based attack called SigAttack. SigAttack extracts the signature of an encryption design and explores the vulnerability. It successfully decrypts many existing encryption schemes such as Zhou's encryption, SARLock and Anti-SAT. We have compared the performance of SigAttack with exact and approximate attacks to demonstrate its efficiency and accuracy.

Acknowledgment

This work is partially supported by NSF under CNS-1441695, CCF-1533656, and CNS-1651695.

References

[1] Y. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *USENIX*, 2007.
[2] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing ic piracy using reconfigurable logic barriers. *IEEE Design and Test*, 27(1), 2010.
[3] S. Dupuis, P. S. Ba, G. D. Natale, M. L. Flottes, and B. Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *IEEE International On-Line Testing Symposium*, pages 49–54, 2014.
[4] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Security analysis of logic obfuscation. In *DAC*, pages 83–89, 2012.
[5] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. Security analysis of integrated circuit camouflaging. In *CCS*, pages 709–720, 2013.
[6] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri. Fault analysis-based logic encryption. *IEEE Transactions on Computers*, 64(2), 2015.
[7] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou. Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks. In *DATE*, pages 85–90, 2018.
[8] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin. AppSAT: Approximately deobfuscating integrated circuits. In *HOST*, pages 95–100, 2017.
[9] Y. Shen, Y. Li, A. Rezaei, S. Kong, D. Dlott, and H. Zhou. Besat: Behavioral sat-based attack on cyclic logic encryption. In *ASP-DAC*, 2019.
[10] Y. Shen, A. Rezaei, and H. Zhou. A comparative investigation of approximate attacks on logic encryptions. In *ASP-DAC*, pages 271–276, 2018.
[11] Y. Shen, A. Rezaei, and H. Zhou. Sat-based bit-flipping attack on logic encryptions. In *DATE*, pages 629–632, 2018.
[12] Y. Shen and H. Zhou. Double dip: Re-evaluating security of logic encryption algorithms. In *GLSVLSI*, pages 179–184, 2017.
[13] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *HOST*, pages 137–143, 2015.
[14] J. B. Wendt and M. Potkonjak. Hardware obfuscation using puf-based logic. In *ICCAD*, pages 270–277, 2014.
[15] Y. Xie and A. Srivastava. Mitigating SAT attack on logic locking. In *CHES*, pages 127–146, 2016.
[16] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte. Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks. In *CHES*, 2017.
[17] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu. SARLock: SAT attack resistant logic locking. In *HOST*, pages 236–241, 2016.
[18] H. Zhou. A humble theory and application for logic encryption. In *Cryptology ePrint Archive, Report 2017/998*, 2017.