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.

1. 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 [4, 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].

With the discovery of the SAT-based attack, many new encryption and decryption techniques are proposed [7, 9, 11]. such as SARLock [17] and Anti-SAT [15] are proposed to ensure that an exponential number of iterations are necessary in the SAT-based attack. However, since error rate of SARlock and Anti-SAT is exponentially low, approximate attacks such as Double DIP [12] and AppSAT [8] are proposed to find a key with extremely low error rate. To accurately measure the performance of approximate attacks, Error-Controllable Encryption [10] is proposed so that error rate of a key value can be exactly calculated. On the other hand, bypass attack [16] can explicitly fix a few errors under an approximate key to get the correct circuit.

With the drawbacks of SARLock and Anti-SAT in mind, Zhou [18] investigated the design space of general logic encryption, and proposed logic encryptions with the best tradeoff between error rate and attack complexity. After proving that the product of an error number, which is defined as the number of mismatches on all inputs for a given key, and attack complexity for any logic encryption can be at most 2^n, he demonstrated a design that can achieve an arbitrary error number M with the attack complexity of $2^n/M$. Therefore, the measure of attack complexity is by the SAT-based attack, specifically, by the number of DIPs (Distinguishing Input Patterns) needed to exclude all wrong keys.

It naturally draws us to this critical question: is the SAT-based attack with DIP 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.

2. RELATED WORKS

2.1 SAT-based Attack

There are various traditional logic encryption techniques. However, these encryption techniques fall into the same situation: a large number of wrong keys can lead to the same wrong input-output pair. Therefore, to find the correct key value, a strategy is to prune out all wrong keys based on the observation of some input-output pairs. To achieve this goal, an attack based on the SAT (satisfiability) solver is developed. As Algorithm 1 shows, a DIP X_i is found in each iteration, and the CNF (Conjunctive Normal Form) of a correct input-output pair $C(X_i, K, Y_i)$ is added to constrain the keys. The algorithm continues until the remaining keys cannot produce new DIPs. Then solving the constraints reveals the correct key.

Algorithm 1 SAT-based Attack

Input: Encrypted circuit $C(X, K, Y)$, and activated circuit $eval$.
Output: Correct key K^*.
1: i = 1
2: $F_1 = C(X, K_1, Y_1) \wedge C(X, K_2, Y_2)$
3: **while** $SAT[F_i \wedge (Y_1 \neq Y_2)]$ **do**
4: $X_i = SAT(F_i \wedge (Y_1 \neq Y_2))$
5: $Y_i = eval(X_i)$
6: $F_{i+1} = F_i \wedge C(X_i, K_1, Y_i) \wedge C(X_i, K_2, Y_i)$
7: $i = i + 1$
8: **end while**
9: $K^* = SAT(F_i)$

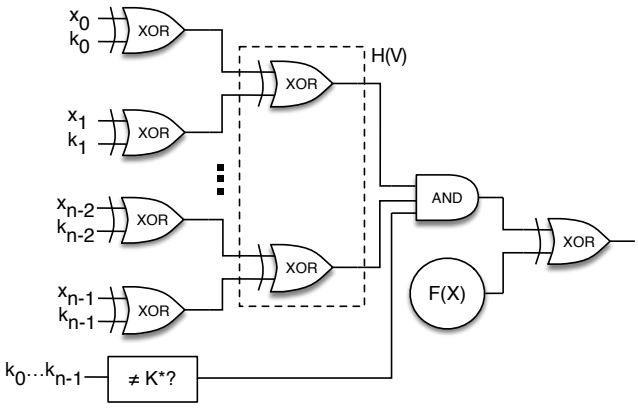

Figure 1: A logic encryption design with both high error number for a wrong key and high attack complexity.

2.2 General Framework for Logic Encryption

Recently, 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 the input [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 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.

3. SIGATTACK: HIGH-LEVEL SAT-BASED ATTACK

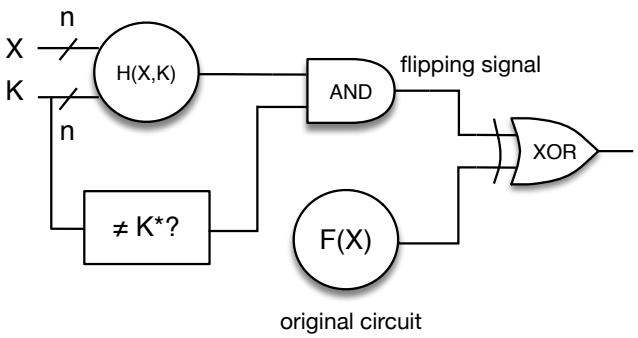

Figure 2: The general scheme that every possible logic encryption is equivalent to.

As shown in Figure 2, the function $H(X, K)$ gives the pattern how the output of the encrypted circuit is different from the original circuit $F(X)$, and the comparison between K and K^* is used to mask off the flipping. 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.

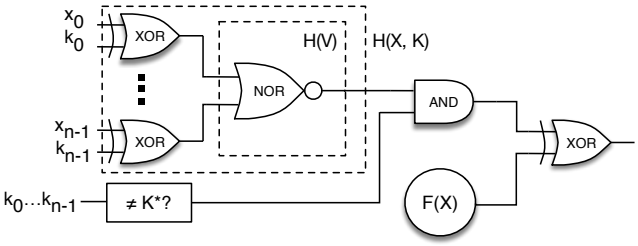

Figure 3: Model SARLock in the general framework and Zhou's encryption.

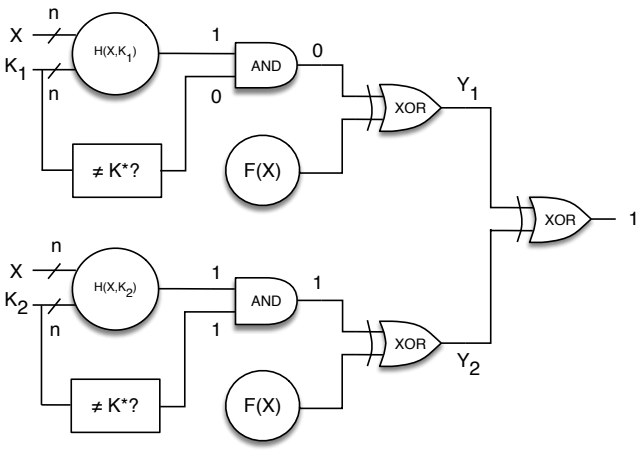

Figure 4: The circuit that reveals the correct key by one SAT query.

The basic idea of all SAT-attack resilient logic encryptions is to make sure that the minimal number of primary inputs to generate at least one $H(X, K) = 1$ for each wrong key is exponential. For example, in SARLock, this is done by making sure that there is only one X for each wrong key such that $H(X, K) = 1$.

In this section, we propose a new high-level SAT-based attack called SigAttack, which extracts a key-revealing signature of a design. Section 3.1 formally defines the signature in logic encryption, and proves that if attackers can extract the signature of an encrypted circuit, the correct key can be revealed. Section 3.2 to Section 3.4 demonstrate that SigAttack can successfully decrypt many existing encryption techniques.

3.1 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 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 as

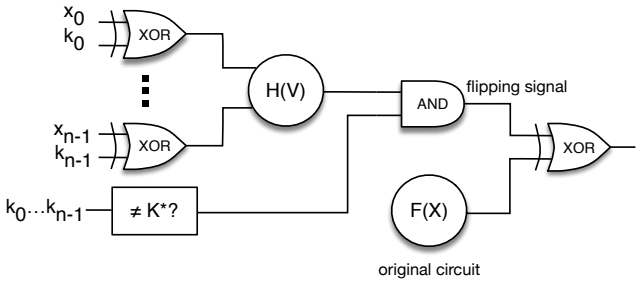

Figure 5: Zhou's encryption.

shown in Figure 4, 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 3.1. *For any given encryption circuit $C(X, K, Y)$, a signature $Sig(X, K)$ is defined as any Boolean expression such that:*

1. $\forall X, K, K \neq K^* : Sig(X, K) \Rightarrow C(X, K, Y) \wedge (Y \neq F(X))$;
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 3.1. *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 $C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge Sig(X, K_1) \wedge Sig(X, K_2) \wedge (Y_1 \neq Y_2)$.*

PROOF. We want to prove that at least one of two keys, K_1 and K_2, is assigned to K^* when the SAT solver handles the proposed query.

Let us assume both K_1 and K_2 are not equal to K^*. As a result, condition 1 in Definition 3.1 indicates that $Y_1 \neq F(X)$ and $Y_2 \neq F(X)$, which leads to $Y_1 = Y_2$ and contradicts the SAT query, and therefore, contradicts condition 2 that ensures a SAT solver can always find such an assignment. Thus, we proved either K_1 or K_2 is the correct key, which can be located by comparing Y_1 and Y_2 with the correct output under X. □

3.2 SigAttack on Zhou's Encryption

Based on the design in Figure 2, a general encryption scheme in Figure 5 has been proposed by Zhou [18], which is to achieve 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 3.1, 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 that 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 that attackers do not know how bits are paired between X and K. In other words, we should explore that 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 that 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 corresponding to k, 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 3.2.

Theorem 3.2. *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).*

PROOF. Assume we split key bits of K into two sets, where K_A contains key bits that the SAT query can be satisfied, and K_B contains the remaining key bits that the SAT query cannot be satisfied. Correspondingly, we can split bits of V for $H(V)$ into two sets, V_A and V_B, respectively.

Assume among n bits in V, there are p bits in V_B, which are $v_0...v_{p-1}$. We want to prove that the outputs of $H(V)$ only depend on bits in V_A, which are $v_p...v_{n-1}$. Therefore, the output of $H(V)$ only depend on $k_p...k_{n-1}$ and $x_p...x_{n-1}$, and key bits in K_B will not affect the output of $H(V)$.

Our algorithm goes through each key bit. For v_0, since the SAT query cannot be satisfied, it indicates that the $H(V)$ only depends on $v_1...v_p...v_{n-1}$, and if $v_1...v_p...v_{n-1}$ are fixed, the output of $H(V)$ is determined. Similarly, from the analysis of v_1, $H(V)$ only depends on $v_0v_2...v_p...v_{n-1}$. Thus, we can conclude that if $v_2...v_p...v_{n-1}$ are fixed, the output of $H(V)$ is determined.

The reason is as follows. Assume outputs of $H(V)$ with $v_0v_1 = 00, 01, 10, 11$ are Y_A, Y_B, Y_C, Y_D respectively. From the v_0 analysis, $Y_A = Y_C$, $Y_B = Y_D$ no matter what the assignments of $v_2...v_{n-1}$ are. Similarly, the v_1 analysis shows that $Y_A = Y_B$, $Y_C = Y_D$. Therefore, $Y_A = Y_B = Y_C = Y_D$, and we have a conclusion that the output of $H(V)$ is determined if $v_2...v_{n-1}$ are fixed.

Hence, from the analysis from v_0 to v_{p-1}, we can conclude that the output of $H(V)$ only depends on bits $v_p...v_{n-1}$. □

Theorem 3.2 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. However, if there is at least one key bit that leads the proposed SAT query cannot be satisfied, the SAT solver has the flexibility to assign different values to that bit, so $H(V) = 1$ for both circuits can be satisfied, and either K_1 or K_2 equals K^*.

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 shown in Algorithm 2. From line 1 to line 8, the pairing between each bit of X and K is found. From line 9 to line 16, we add constraints to a SAT solver to guarantee that outputs of $H(V)$ for both circuits are one. V_1 and V_2 in line 9 can be found by SAT queries and comparing SAT assignments of bits in each found pair. If the second assignment V_2 cannot be found, we conduct bypass attack. Line 5 indicates that either $\widehat{Y_1}$ or $\widehat{Y_2}$ can be assigned to the correct value since they are symmetric, and the bracket from line 12 to line 15 indicates that we should constrain both $C(X, K_1, Y_1)$ and $C(X, K_2, Y_2)$ based on V_1 and V_2, respectively. v_i^j, x_i^j and k_i^j denote the ith bit of V, X and K in the jth copy of the circuit C. As a result, we can prove the following theorem:

Algorithm 2 SigAttack on Zhou's Encryption

Input: Encrypted circuit $C(X, K, Y)$, and activated circuit $eval$.
Output: Correct key K^*.
1: **for** each key input bit $k \in K$ **do**
2: $\widehat{X}, \widehat{K_1}, \widehat{K_2}, \widehat{Y_1}, \widehat{Y_2} = SAT(C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge (Y_1 \neq Y_2) \wedge (k_1 \neq k_2) \wedge ((K_1 \backslash k_1) = (K_2 \backslash k_2)))$
3: **for** each not paired input bit $x \in X$ **do**
4: $X' = \widehat{X}$ with x flipped
5: **if** $((\widehat{Y_1} = eval(\widehat{X})) \wedge C(X', \widehat{K_1}, Y_1') \wedge (\widehat{Y_2} = Y_1')) \vee ((\widehat{Y_2} = eval(\widehat{X})) \wedge C(X', \widehat{K_2}, Y_2') \wedge (\widehat{Y_1} = Y_2'))$ **then**
6: Pair.add(x, k)
7: **end for**
8: **end for**
9: Use SAT queries to find V_1, V_2 s.t. $(V_1 \neq V_2) \wedge H(V_1) \wedge H(V_2)$
10: $F = C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge (Y_1 \neq Y_2)$
11: **for** each pair $(x_i, k_i) \in Pair$ **do**
12: **if** $v_i^1(v_i^2)$ in $V_1(V_2) = 0$ **then**
13: $F = F \wedge (x_i^1(x_i^2) = k_i^1(k_i^2))$
14: **if** $v_i^1(v_i^2)$ in $V_1(V_2) = 1$ **then**
15: $F = F \wedge (x_i^1(x_i^2) \neq k_i^1(k_i^2))$
16: **end for**
17: $K^* = SAT(F)$
18: **if** $\nexists V_2$ **then**
19: K^* = bypass_attack($C(X, K, Y)$, $eval$)

Theorem 3.3. *Zhou's encryption shown in Figure 5 can be attacked by Algorithm 2 if Property 1 holds.*

3.3 SigAttack on SARLock

The analysis from Section 3.2 indicates that Algorithm 2 can be used to 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 Algorithm 2 on this design. From line 1 to line 8, we can pair each bit x_i and k_i for i from 0 to $n-1$. However, in Line 9, 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].

3.4 SigAttack on Anti-SAT

We model an Anti-SAT design [15] to Zhou's encryption as shown in Figure 6, and $H(V)$ is the original Anti-SAT blocks.

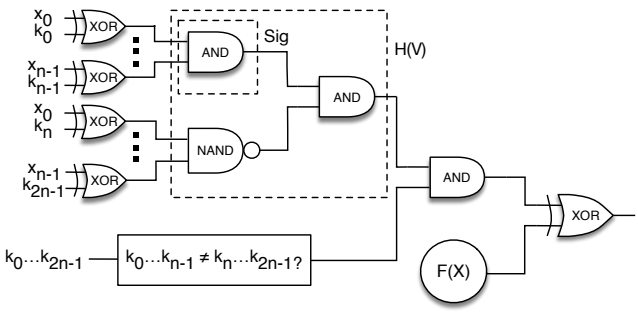

Figure 6: Model Anti-SAT in Zhou's encryption.

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 $k_nk_{n+1}...k_{2n-1}$.

Since Figure 6 is not exactly fit for the model of Zhou's encryption, could we still attack it by SigAttack? The answer is yes. Following the Algorithm 2, 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 the SAT query in line 17 of Algorithm 2. 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 6) 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.

4. 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 6, 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 the design of Zhou's encryption shown in Figure 1. *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,

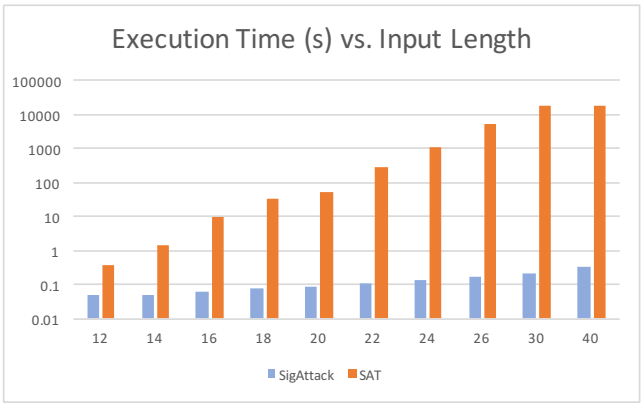

Figure 7: Execution time of performing SigAttack on Zhou's Encryption compared with the SAT-based attack.

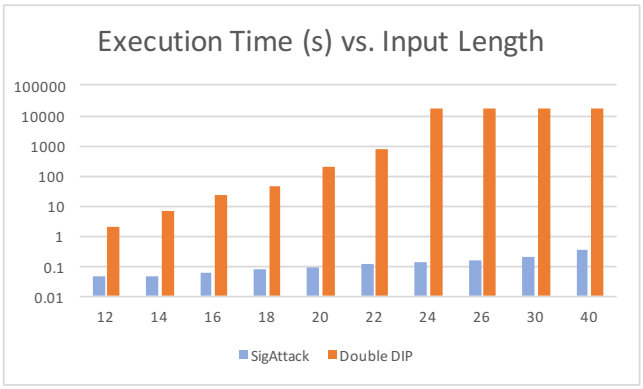

Figure 8: Execution time of performing SigAttack on Zhou's Encryption compared with Double DIP.

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 7 and 8 demonstrate the relation between the execution time and 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.

We perform sigAttack on SARLock and Anti-SAT shown in Figure 3 and Figure 6. As we analyzed in Section 3.3, 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 9 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.

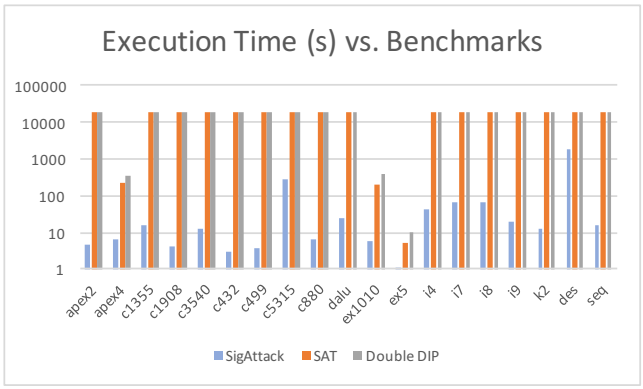

Figure 9: Execution time of performing SigAttack on Anti-SAT compared with the SAT-based attack and Double DIP.

5. 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 5. 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 3 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 3.2 to Section 3.4, 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.

6. 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.

The development of SigAttack inspires us to find countermeasures so that the signature is hard to be extracted for the future logic encryption design. One advice is that to propagate the correctness of the key to the output of an encrypted circuit, a part of key bits have to be fixed. Therefore, the SAT solver loses the flexibility to assign correct values to the key.

Acknowledgment

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

7. REFERENCES

[1] Yousra Alkabani and Farinaz Koushanfar. Active hardware metering for intellectual property protection and security. In *USENIX*, pages 20:1–20:16, 2007.

[2] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. Preventing ic piracy using reconfigurable logic barriers. *IEEE Design and Test*, 27(1), 2010.

[3] Sophie Dupuis, Papa-Sidi Ba, Giorgio Di Natale, Marie-Lise Flottes, and Bruno Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *IOLTS*, pages 49–54, 2014.

[4] Jeyavijayan Rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of logic obfuscation. In *DAC*, pages 83–89, 2012.

[5] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of integrated circuit camouflaging. In *CCS*, 2013.

[6] Jeyavijayan Rajendran, Huan Zhang, Chi Zhang, Garrett S. Rose, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. Fault analysis-based logic encryption. *IEEE Transactions on Computers*, 64(2), 2015.

[7] Amin Rezaei, Yuanqi Shen, Shuyu Kong, Jie Gu, and Hai Zhou. Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks. In *DATE*, pages 85–90. IEEE, 2018.

[8] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. AppSAT: Approximately deobfuscating integrated circuits. In *HOST*, pages 95–100, 2017.

[9] Yuanqi Shen, You Li, Amin Rezaei, Shuyu Kong, David Dlott, and Hai Zhou. Besat: Behavioral sat-based attack on cyclic logic encryption. In *ASP-DAC*, 2019.

[10] Yuanqi Shen, Amin Rezaei, and Hai Zhou. A comparative investigation of approximate attacks on logic encryptions. In *ASP-DAC*, 2018.

[11] Yuanqi Shen, Amin Rezaei, and Hai Zhou. Sat-based bit-flipping attack on logic encryptions. In *DATE*, pages 629–632. IEEE, 2018.

[12] Yuanqi Shen and Hai Zhou. Double dip: Re-evaluating security of logic encryption algorithms. In *GLSVLSI*, pages 179–184, 2017.

[13] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In *HOST*, pages 137–143, 2015.

[14] James B Wendt and Miodrag Potkonjak. Hardware obfuscation using puf-based logic. In *ICCAD*, pages 270–277. IEEE Press, 2014.

[15] Yang Xie and Ankur Srivastava. Mitigating SAT attack on logic locking. In *CHES*, pages 127–146, 2016.

[16] Xiaolin Xu, Bicky Shakya, Mark M Tehranipoor, and Domenic Forte. Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks. In *CHES*, 2017.

[17] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan J V Rajendran, and Ozgur Sinanoglu. SARLock: SAT attack resistant logic locking. In *HOST*, pages 236–241, 2016.

[18] Hai Zhou. A humble theory and application for logic encryption. In *Cryptology ePrint Archive, Report 2017/998*, 2017.