Double DIP: Re-Evaluating Security of Logic Encryption Algorithms

Yuanqi Shen
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208
yuanqishen2020@u.northwestern.edu

Hai Zhou
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208
haizhou@eecs.northwestern.edu

ABSTRACT

Logic encryption is a hardware security technique that uses extra key inputs to lock a given combinational circuit. A recent study by Subramanyan et al. shows that all existing logic encryption techniques can be successfully attacked. As a countermeasure, SARLock was proposed to enhance the security of existing logic encryptions. In this paper, we re-evaluate the security of these approaches. A SAT-based attack called Double DIP is proposed and shown to successfully defeat SARLock-enhanced encryptions.

1. INTRODUCTION

The active participation of external entities in the design and manufacturing of ICs has produced numerous hardware security issues. Among all the hardware security problems, the counterfeiting, piracy, and unauthorized overproduction of electronic components have become a major challenge for government and industry [6, 14]. Most leading-edge design houses have outsourced their fabrication to the offshore foundries for the sake of lower labor and manufacturing cost. However, many offshore foundries are hard to be trusted since they may be in a country without consummate enforcement law for IP protection [13]. The economic impacts and security hazards of hardware piracy are not apt to be neglected compared to software, but is even more severe. The loss due to global hardware piracy has now reached the level of billions per month, with a major share in almost all electronic devices [1]. It was reported by the Alliance for Gray Market and Counterfeit Abatement that about 10% of the start-of-the-art technology products available on market are counterfeits [6].

Logic encryption is a technique proposed to thwart counterfeiting, piracy, and unauthorized overproduction of electronic components [3, 7–10]. It inserts extra gates called key gates into IC design to hide its original functionality. The key inputs are connected to a tamper-proof memory, and the IC only produces all correct input-output pairs if key inputs are correct key values. In that case, even though the foundry is able to access the netlist, and attackers can either steal the netlist from the foundry or reverse engineering the netlist from layout and mask information [12], they will not get functional circuit without loading the correct key value.

Various logic encryption techniques have been exploited. Rajendran et al. [9] propose a logic encryption algorithm that inserting XOR/XNOR gates and Multiplexers based on fault analysis. Dupuis et al. [4] propose a rare value based logic encryption technique and insert AND/OR gates to balance the probability of a signal between 0 and 1. Wendt et al. [15] use multiplexers to select paths in the netlist, and the signal of selection depends on the output of a PUF. The input of the PUF is counted as key inputs. Rajendran et al. [8] analyze the netlist and carefully insert the key gates by assigning weights to key gates. Alkabani et al. [2] replicate a few states of the finite state machine (FSM), and key values control the flow of state transitions.

It should be mentioned that after the procedure of logic encryption, the inserted key gates can be further obfuscated so that it is hard for untrusted foundry and attackers to directly remove them from the netlist [5].

However, almost all existing (combinational) logic encryptions techniques have been decrypted by a SAT-based attack proposed by Subramanyan et al. [11]. It utilizes advanced SAT solver to narrow down the scope of correct key values. Then a work by Yasin et al. called SARLock successfully thwarts SAT-based decryption algorithm by rendering the attack effort exponential in the number of bits in the secret key [17].

This paper develops a new SAT-based decryption technique called Double DIP, which can be used to attack SARLock technique. Contributions of this paper are as follows:

1. We present a new logic decryption algorithm called Double DIP. Double DIP excludes at least two wrong keys each iteration, ensuring wrong keys in the part of traditional logic encryption being excluded without taking exponential iterations.

2. We evaluate the correctness and efficiency of Double DIP comparing with SAT attack. Double DIP takes a small number of iterations to find key values K, and the encrypted circuit with K will behave the same as the correct one except for one or very limited numbers of inputs.

3. If traditional logic encryption key value K_1 is not unique, Double DIP may take similar number of iter-

GLSVLSI '17, May 10-12, 2017, Banff, AB, Canada.

DOI: http://dx.doi.org/10.1145/3060403.3060469

ations as SAT attack. However, we demonstrate that Double DIP can still efficiently thwart SARLock technique if K_1 and SARLock key K_2 can be separated.

The rest of the paper is organized as follows. Section 2 provides a brief overview of SAT attack and SARLock. Section 3 describes the mechanism of Double DIP and why Double DIP can be treated as a successful attack. Section 4 compares the efficiency of SAT attack and Double DIP to solve benchmarks encrypted by the combination of traditional logic encryption and SARLock technique. Section 5 discusses the influence if the key of traditional logic encryption is not unique, and Section 6 concludes the paper.

2. PRELIMINARIES

In this section, we will introduce SAT attack and SARlock techniques. SAT attack is a SAT-based technique to attack logic encryptions [11]. SARLock is a logic encryption enhancement against SAT attack [17], and the SAT attack needs an exponential number of iterations to exclude all wrong keys after the circuit is encrypted by SARLock.

2.1 SAT attack

As we already discussed in the previous section, an encryption of a circuit is to modify the circuit into another one with some extra key inputs such that the input-output relation is the same as the original one only when the correct key value is applied. Fig. 1 presents a simple logic encryption example. Fig. 1(a) is an original circuit with AND/OR/XOR gates. Fig. 1(b) inserts AND/OR key gates into netlist, and correct key inputs value is 01. Fig. 1(c) inserts XOR/XNOR key gates and correct key inputs value is 10.

The attack model assumes that the logic of modified circuit (denoted as *locked circuit*) is known, and the original circuit could be bought and accessed as a black-box. A recent work by Subramanyan et al. [11] has attracted lots of attention in hardware security. They proposed a SAT attack to (combinational) logic encryptions, and found that almost all of encrypted circuits of all existing logic encryption approaches [3, 4, 8–10] have been corrupted by their approach. The SAT attack works as Algorithm 1.

Algorithm 1 SAT Attack Algorithm

Input: C and $eval$.
Output: K_c.
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: $X_i = sat\_assignment_X(F_i \wedge (Y_1 \neq Y_2))$
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: $K_c = sat\_assignment_{K1}(F_i)$

It iteratively finds the assignment to the following CNF (Conjunctive Normal Form) until it is unsatisfiable:

$$C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge (Y_1 \neq Y_2),$$

where $C(X, K, Y)$ is the CNF of the locked circuit with input X, key K, and output Y. Each time when X_i as an assignment of X is generated, its corresponding output Y_i from the original circuit is found, and they are used to further constrain K_1 and K_2 by adding

$$C(X_i, K_1, Y_i) \wedge C(X_i, K_2, Y_i)$$

to the existing CNF. The X_i generated in each iteration is called *DIP* (Differentiating Input Pattern), since it is the input that differentiates two possible keys under existing constraints. The iteration will stop when the CNF is no longer satisfiable, which means that there exists no input that can differentiate possible keys. Therefore, any key that satisfies the current constraints is the correct key, which can be computed by SAT on the constraints.

After the publication of the SAT attack on logic encryption, quickly there were many approaches being proposed to enhance logic encryption against the SAT attack. Ideas include either to increase the complexity of the locked circuit such that finding a DIP cannot be easily solved by SAT, or to increase the number of iterations in the process. There is no solid proposal in the first direction, since even though SAT is in general NP-hard, creating a hard instance is generally an unsolved problem.

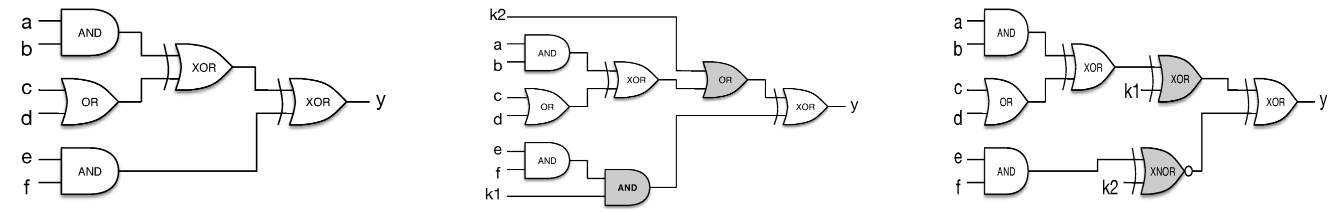
One of the reasons that the SAT attack has been successful is that it needs to use only a small number of DIPs to exclude all wrong keys for a locked circuit. This means that some DIPs in the iterations exclude a substantial number of wrong keys. Therefore, in the second direction, one way to increase the number of necessary iterations in SAT attack is to make sure that there are substantially large number of wrong keys requesting similarly large number of DIPs to exclude.

2.2 SARLock

SARLock is a logic encryption enhancement against SAT attack proposed by Yasin et al. [17]. The idea of SARLock is to make sure that each wrong key can only be excluded by one DIP. Therefore, the SAT attack needs an exponential number of DIPs to exclude all wrong keys. The simplest design is to have the output flipped only when the key is equal to the input, unless the key is the correct one. As we can see, only the same input can exclude a given wrong key.

Of course the simple design might be easily broken by an attacker, for example, by just picking a random key and flipping the output when the input is the same as the key. To secure against this, they proposed to add the SARLock with a key K_2 on top of any traditional logic encryption with a key K_1. They also proposed to scramble K_1 and K_2 together (e.g., by XORing them) and then apply the simple SARLock on it. Fig. 2 shows the schematic of the combination of traditional logic encryption and SARLock. The circuit is initially encrypted by a traditional logic encryption with a key K_1. Then K_1 is scrambled with SARLock key inputs K_2, and the scrambled result is compared with inputs to generate a flip signal, which is used to flip the output of encrypted circuit when they are equal. This flip signal is 1 when inputs are equal to the scrambled result of K_1 and K_2, and this scrambled result is not equal to the scrambled result of correct K_1 and correct K_2. The mask guarantees when K_1 and K_2 are correct key values, the output of encrypted circuit will not be flipped.

As we can see, the SARLock part can ensure that the SAT attack needs to take exponential time while the traditional logic encryption part can ensure security against simple guess attacks.



(a) Original Circuit with AND/OR/XOR gates.

(b) Encrypted circuit with AND/OR key gates. Correct key is 01.

(c) Encrypted circuit with XOR/XNOR key gates. Correct key is 10.

Figure 1: Logic encryption example.

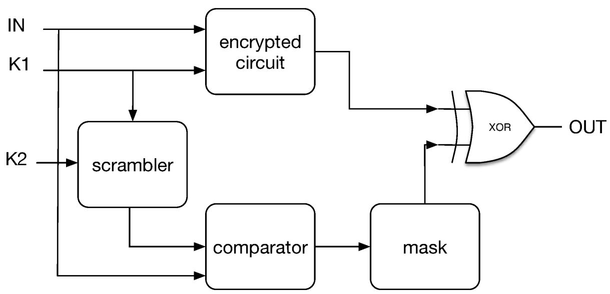


Figure 2: Traditional logic encryption + SARLock.

3. DOUBLE DIP: A NEW ATTACK

Can we then say that this combined SARLock is secure now? Hardly so, since we have developed a new attack approach nick-named *Double DIP* to corrupt the combined SARLock. The ideas can be described as follows. Double DIP is an extension of the SAT attack, whose main idea is to iteratively find DIPs (differentiating input pattern) to exclude more and more wrong keys. Instead of finding a DIP, Double DIP will find 2DIP (doubly differentiating input pattern) in each iteration by finding SAT assignment for the following CNF:

$$C(X,K_1,Y_1) \wedge C(X,K_2,Y_2) \wedge C(X,K_3,Y_1) \wedge$$
$$C(X,K_4,Y_2) \wedge (Y_1 \neq Y_2) \wedge (K_1 \neq K_3) \wedge (K_2 \neq K_4),$$

where $C(X,K,Y)$ is the locked circuit with input X, key K, and output Y.

If the CNF is satisfiable, the input assignment X_i will be used to find the corresponding Y_i in the original black-box circuit, and the following constraint will be added to the CNF:

$$C(X_i,K_1,Y_i) \wedge C(X_i,K_2,Y_i)$$
$$\wedge C(X_i,K_3,Y_i) \wedge C(X_i,K_4,Y_i).$$

The algorithm of Double DIP is shown in Algorithm 2. In the algorithm, a 2DIP can exclude at least two wrong keys, ensuring wrong keys in the traditional logic encryption part being excluded. When the iterations stop, meaning the constrained CNF is not satisfiable, we will find a key K that satisfies all the existing constraints $\forall_{i=1}^{n} C(X_i, K, Y_i)$.

It should be mentioned that Double DIP can be further extended to K DIP, which excludes k wrong keys in each iteration. However, the increasing of k leads to an increasing number of clauses in SAT solver, which takes SAT solver more execution time to find an assignment in each iteration.

Algorithm 2 Double DIP

Input: C and $eval$.
Output: K_c.
i = 1
$F_1 = C(X,K_1,Y_1) \wedge C(X,K_2,Y_2) \wedge C(X,K_3,Y_1) \wedge C(X,K_4,Y_2)$
while $sat[F_i \wedge (Y_1 \neq Y_2) \wedge (K_1 \neq K_3) \wedge (K_2 \neq K_4)]$ **do**
  $X_i = sat\_assignment_X(F_i \wedge (Y_1 \neq Y_2) \wedge (K_1 \neq K_3) \wedge (K_2 \neq K_4))$
  $Y_i = eval(X_i)$
  $F_{i+1} = F_i \wedge C(X_i,K_1,Y_i) \wedge C(X_i,K_2,Y_i) \wedge C(X_i,K_3,Y_i) \wedge C(X_i,K_4,Y_i)$
  $i = i + 1$
end while
$K_c = sat\_assignment_{K1}(F_i)$

Theorem 3.1. *When applied on the same encrypted circuit, there always exists a SAT configuration such that SAT attack has at least the same number of iterations as Double DIP. When the Double DIP terminates, the key of traditional logic encryption K_1 is guaranteed to be correct.*

Assume $(X_0,Y_0),(X_1,Y_1),...,(X_i,Y_i)$ are input-output pairs that Double DIP finds from the beginning to the termination. Since Double DIP adds extra constraints into constraints of SAT attack, (X_0,Y_0), (X_1,Y_1), ..., (X_i,Y_i) are also input-output pairs that satisfy constraints of SAT attack. When Double DIP and SAT attack apply on the same encrypted circuit, they can have the same constraints of input-output pairs (X_0,Y_0), (X_1,Y_1), ..., (X_i,Y_i) after i+1 iterations. If DIP X that can exclude only one wrong key exists, SAT attack takes extra iterations to add (X,Y), which is a pair of X and the corresponding output Y in the original circuit, into constrained CNF until such X cannot be found. However, if such X does not exist, SAT attack and Double DIP take the same number of iterations. So when applied on the same encrypted circuit, there always exists a SAT configuration such that SAT attack has at least the same number of iterations as Double DIP.

Then we prove when the Double DIP terminates, the key of traditional logic encryption K_1 is guaranteed to be correct. Assume K_1 is extended to $K_1 \cdot K_2$ by the SARLock technique. Proving K_1 is correct when the Double DIP terminates is equivalent to prove that the Double DIP does not terminate when K_1 is not correct. Then we prove by contradiction. In traditional logic encryption, K_1^c is a correct key, and K_1^{inc} is one of the wrong keys. K_2^{inc1} and K_2^{inc2} are two of the incorrect keys in SARlock. Assume the K_1 is not correct, but the Double DIP terminates. Then we have an assignment $K_1 = K_1^c \cdot K_2^{inc1}$, $K_2 = K_1^{inc} \cdot K_2^{inc1}$, $K_3 = K_1^c \cdot K_2^{inc2}$ and $K_4 = K_1^{inc} \cdot K_2^{inc2}$ to meet the satisfiability. Then the Double DIP does not terminate, which is contradict to our assumption.

It should be noted here that the key K found by the Double DIP cannot be guaranteed to be the correct one. This is because the existing constraints, when combined with the DIP CNF in the original SAT attack, may still be satisfiable. However, even in this case, since the 2DIP CNF, when combined with the existing constraints, is not satisfiable, the found key K can be wrong on at most one input.

Now we will argue that, if a key is wrong on at most one input or even a few inputs, it can be treated as a successful attack. Firstly, when an attacker plugs in this key in the locked circuit, the circuit will behave the same as the correct one except for one or very limited numbers of inputs. The chance to excite the error is exponentially small. Even the attacker sells this circuit to the market, the chance for the market to detect it is similarly small. Secondly, even if an error is discovered by luck, it can be easily corrected by hardwiring the correct output for that specific input. Since it is guaranteed to have only one or a few errors, such a correction is much better than finding a new key by adding the new input pattern.

The last, but not the least, justification comes from a solid theoretical foundation. Statistical techniques are commonly used in cryptography. In cryptography, the information is secure if there does not exist a polynomial time algorithm that can compute the same information with high probability. Based on this definition, we can see that our Double DIP attack has achieved the correctness since the key it finds will produce the correct results with very high probability.

4. EVALUATION

Now we evaluate the effectiveness of Double DIP. The benchmarks are from Subramanyan et al. [11], which are encrypted combinational circuits by a robust encryption technique proposed by Dupuis et al. [4], and these encrypted circuits are originally encrypted on circuits from the Microelectronics Center of North Carolina (MCNC). Dupuis et al. [4] is an algorithm that computes the probabilities of all signals and carefully inserts AND and OR gates so that rare values are minimized. We choose each benchmark with an area overhead of encryption in 5%, 10% and 25%. Then we further encrypt circuits with SARLock technique for the purpose of evaluation. Bits of SARLock keys K_2 are set to be equal with bits of inputs for the convenience of comparison.

The Figure 3-5 show the time to decrypt Dupuis et al. [4] + SARLock for both SAT attack and Double DIP. The comparison shows the Double DIP dramatically decreases the execution time since it avoids solving K_2 with exponential iterations. The execution time of Double DIP for most benchmarks is less than 10 seconds, however the SAT attack cannot solve most of benchmarks within our time limit, an hour.

The Table 1 illustrates how many DIPs are needed to solve the benchmark. For most cases, Double DIP takes less than 100 DIPs to finalize the key. However, the SAT attack needs to take exponential iterations. The SAT attack may handle

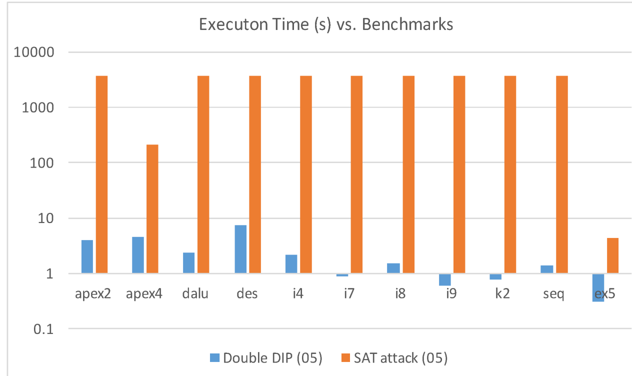


Figure 3: Evaluating Double DIP and SAT attack on benchmarks encrypted with Dupuis et al. [4] (5% area overhead) + SARLock.

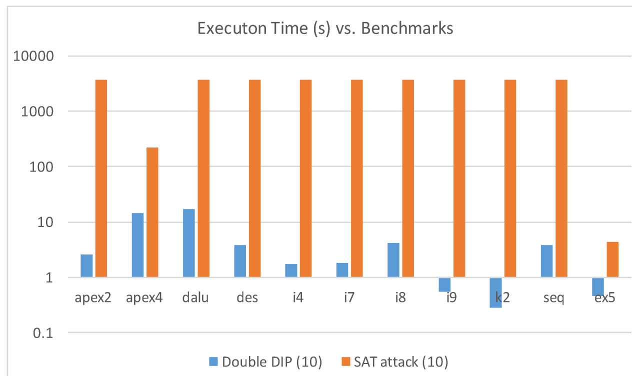


Figure 4: Evaluating Double DIP and SAT attack on benchmarks encrypted with Dupuis et al. [4] (10% area overhead) + SARLock.

the SARLock technique if the number of K_2 is relatively small, but with the increasing of bits of K_2, only Double DIP can solve K_1 efficiently.

The Figure 6-8 compare the execution time of Double DIP to decrypt Dupuis et al. [4] + SARLock and the execution time of SAT attack to decrypt Dupuis et al. [4]. Even though SARLock is further implemented, Double DIP can still efficiently decrypt most of benchmarks with slightly extra execution time.

5. DISCUSSION

During the experiment, surprisingly we found that for some combinations of benchmarks and logic encryption techniques, the Double DIP still took exponential iterations to stop. The reason is that for some logic encryption techniques, more than one correct K_1 is allowed. If correct K_1 is not unique and K_1^{c1} and K_1^{c2} are two of correct K_1 values, then for each iteration, the SAT solver can always find an assignment $K_1 = K_1^{c1} \cdot K_2^c, K_2 = K_1^{c1} \cdot K_2^{inc}, K_3 = K_1^{c2} \cdot K_2^c$ and $K_4 = K_1^{c2} \cdot K_2^{inc}$, and a DIP (X_i, Y_i) may prune K_2 and K_4. In that case, only the combination of one correct K_1 and incorrect K_2 can be pruned, and necessary DIPs are exponential to the bits of K_2.

It leads to an interesting direction to match Double DIP with different logic encryption techniques with several cor-

Table 1: **Bits of keys and required iterations for SAT attack and Double DIP.**

circuits	# of K_2	5% overload			10% overload			25% overload		
		# of K_1	Double DIP	SAT	# of K_1	Double DIP	SAT	# of K_1	Double DIP	SAT
apex2	39	32	56	7106	65	50	8594	162	30	8966
apex4	10	269	44	1022	537	87	1022	1343	98	1022
dalu	75	119	20	4879	237	119	5453	593	1526	4829
des	256	336	12	3046	673	5	2982	1682	11	2795
i4	192	27	9	3601	53	17	7596	133	17	7286
i7	199	76	5	6071	151	12	5547	379	19	5189
i8	133	130	11	5294	260	33	5155	649	28	4874
i9	88	56	3	7552	112	3	6440	281	6	6074
k2	46	93	16	6960	186	2	6970	465	5	6510
seq	41	178	18	5314	356	35	5481	890	366	5149
ex5	8	53	14	254	106	18	254	266	254	254

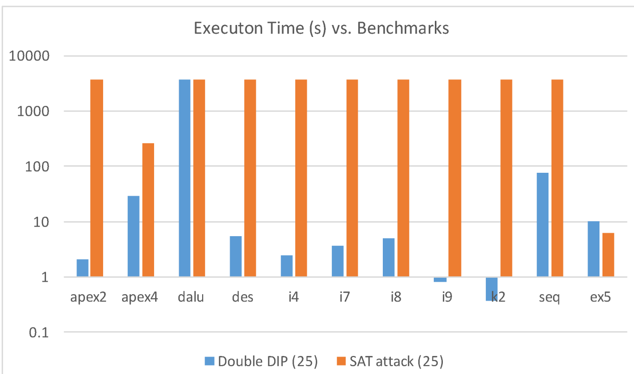


Figure 5: Evaluating Double DIP and SAT attack on benchmarks encrypted with Dupuis et al. [4] (25% area overhead) + SARLock.

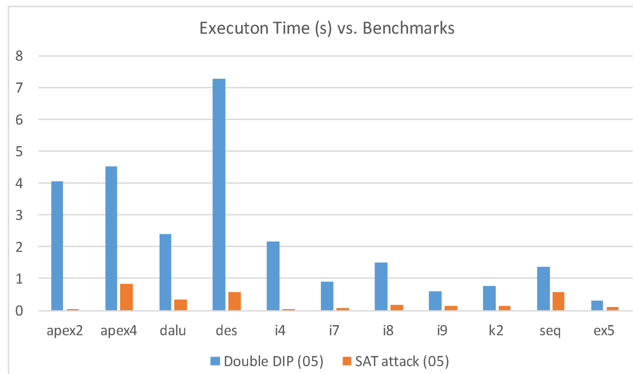


Figure 6: Evaluating Double DIP on benchmarks encrypted with Dupuis et al. [4] (5% area overhead) + SARLock and SAT Attack on benchmarks encrypted with Dupuis et al. [4] (5% area overhead).

rect K_1 values. One of possible solutions will use statistical techniques to separate K_1 and K_2, and set K_2 as a random fixed number. In that case, Double DIP still can find correct K_1 without exponential iterations. It should be mentioned that SAT attack cannot solve correct K_1 value by setting K_2 as a fixed number since the K_2 is most likely to be incorrect, and SAT attack guarantees the final key K is correct for all input-output pairs.

An experiment is performed on the benchmarks from Section 4. We choose Rajendran et al. [8] as traditional logic encryption technique. Each benchmark is encrypted with Rajendran et al. [8] and SARLock, and assume we could separate traditional logic encryption key K_1 and SARLock technique key K_2. We evaluate the Double DIP with and without setting K_2 as zero. The Figure 9-10 illustrate that with a fixed K_2, Double DIP could efficiently solve K_1 of most of benchmarks within 100 seconds. However, if K_2 is not fixed, K_1 of most of benchmarks cannot be solved within an hour.

6. CONCLUSION

In this paper, we propose Double DIP to evaluate the security of the combination of traditional logic encryption and SARLock. SARLock minimizes the efficiency of SAT attack by exponentially increasing the required number of distinguishing input patterns, and only one incorrect key can be pruned each iteration. To avoid exponential iterations, Double DIP only consider incorrect keys causing more than one incorrect input-output pairs, which disables the SARLock and gets the correct key for traditional logic encryption.

We argue that if a key is wrong on at most one input or even a few inputs, it can be treated as a successful attack. The evaluation demonstrates that Double DIP can efficiently thwart SARLock technique if traditional logic encryption key K_1 is unique. We also show that if K_1 is not unique and k_1 and K_2 can be separated, Double DIP can still solve correct K_1 quickly by setting K_2 as a random fixed number.

Our future work involves two aspects. First, proposing new logic decryption approach to attack a state-of-art logic encryption scheme called Anti-SAT [16] since Double DIP still takes exponential iterations to decipher the correct key of Anti-SAT [16]. Second, implementing Double DIP on logic encryption techniques which keys are not unique. Statistical analysis would be a good start to separate K_1 and K_2.

7. ACKNOWLEDGMENT

This work is partially supported by NSF under CNS-1651695 and CNS-1441695, and by SRC under 2014-TS-2559.

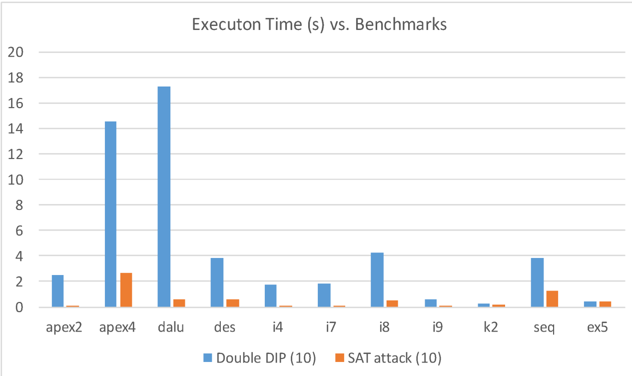


Figure 7: Evaluating Double DIP on benchmarks encrypted with Dupuis et al. [4] (10% area overhead) + SARLock and SAT Attack on benchmarks encrypted with Dupuis et al. [4] (10% area overhead).

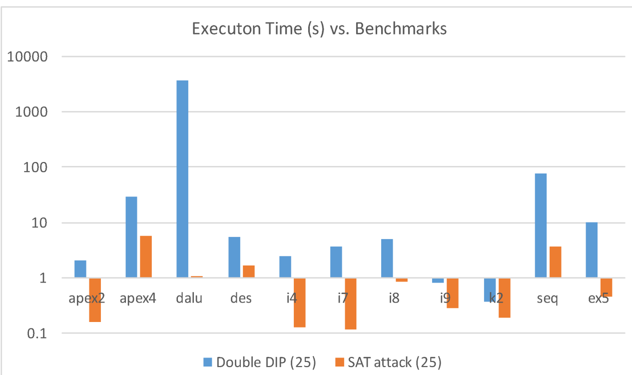


Figure 8: Evaluating Double DIP on benchmarks encrypted with Dupuis et al. [4] (25% area overhead) + SARLock and SAT Attack on benchmarks encrypted with Dupuis et al. [4] (25% area overhead).

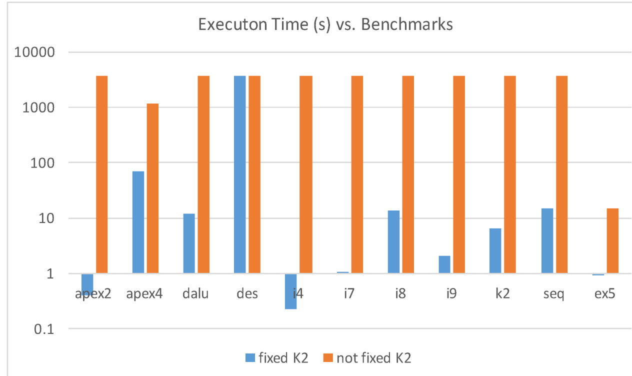


Figure 9: Evaluating Double DIP on benchmarks (Rajendran et al. [8] (5% area overhead) + SARLock) with and without fixed K2.

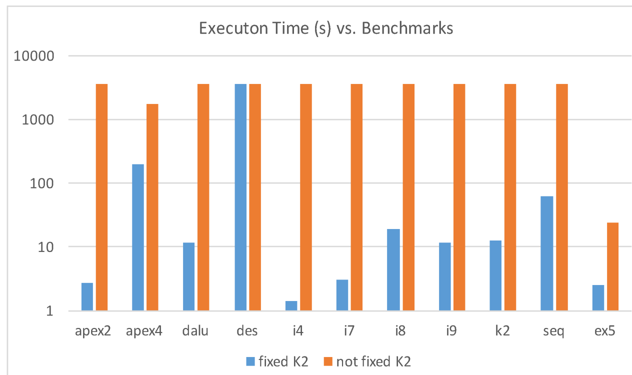


Figure 10: Evaluating Double DIP on benchmarks (Rajendran et al. [8] (10% area overhead) + SARLock) with and without fixed K2.

8. REFERENCES

[1] White paper: The value and management of intellectual assets. http://www.vsi.org.

[2] ALKABANI, Y., KOUSHANFAR, F., AND POTKONJAK, M. Remote activation of ics for piracy prevention and digital right management. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design* (2007), IEEE Press, pp. 674–677.

[3] BAUMGARTEN, A. C. Preventing integrated circuit piracy using reconfigurable logic barriers.

[4] DUPUIS, S., BA, P.-S., DI NATALE, G., FLOTTES, M.-L., AND ROUZEYRE, B. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)* (2014), IEEE, pp. 49–54.

[5] LI, L., AND ZHOU, H. Structural transformation for best-possible obfuscation of sequential circuits. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on* (2013), IEEE, pp. 55–60.

[6] PECHT, M., AND TIKU, S. Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics. *IEEE spectrum 43*, 5 (2006), 37–46.

[7] RAJENDRAN, J., PINO, Y., SINANOGLU, O., AND KARRI, R. Logic encryption: A fault analysis perspective. In *Proceedings of the Conference on Design, Automation and Test in Europe* (2012), EDA Consortium, pp. 953–958.

[8] RAJENDRAN, J., PINO, Y., SINANOGLU, O., AND KARRI, R. Security analysis of logic obfuscation. In *Proceedings of the 49th Annual Design Automation Conference* (2012), ACM, pp. 83–89.

[9] RAJENDRAN, J., ZHANG, H., ZHANG, C., ROSE, G. S., PINO, Y., SINANOGLU, O., AND KARRI, R. Fault analysis-based logic encryption. *IEEE Transactions on Computers 64*, 2 (2015), 410–424.

[10] ROY, J. A., KOUSHANFAR, F., AND MARKOV, I. L. Epic: Ending piracy of integrated circuits. In *Proceedings of the conference on Design, automation and test in Europe* (2008), ACM, pp. 1069–1074.

[11] SUBRAMANYAN, P., RAY, S., AND MALIK, S. Evaluating the security of logic encryption algorithms. In *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on* (2015), IEEE, pp. 137–143.

[12] TORRANCE, R., AND JAMES, D. The state-of-the-art in ic reverse engineering. In *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 363–381.

[13] TRIMBERGER, S. Trusted design in fpgas. In *Proceedings of the 44th annual Design Automation Conference* (2007), ACM, pp. 5–8.

[14] VILLASENOR, J., AND TEHRANIPOOR, M. Chop shop electronics. *IEEE Spectrum 50*, 10 (2013), 41–45.

[15] WENDT, J. B., AND POTKONJAK, M. Hardware obfuscation using puf-based logic. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design* (2014), IEEE Press, pp. 270–277.

[16] XIE, Y., AND SRIVASTAVA, A. Mitigating sat attack on logic locking. In *International Conference on Cryptographic Hardware and Embedded Systems* (2016), Springer, pp. 127–146.

[17] YASIN, M., MAZUMDAR, B., RAJENDRAN, J. J., AND SINANOGLU, O. Sarlock: Sat attack resistant logic locking. In *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on* (2016), IEEE, pp. 236–241.