A Comparative Investigation of Approximate Attacks on Logic Encryptions

Yuanqi Shen, Amin Rezaei, and Hai Zhou
Northwestern University
yuanqishen2020@u.northwestern.edu, me@aminrezaei.com, haizhou@northwestern.edu

ABSTRACT

Logic encryption is an important hardware protection technique that adds extra keys to lock a given circuit. With recent discovery of the effective SAT-based attack, new enhancement methods such as SARLock and Anti-SAT have been proposed to thwart the SAT-based and similar exact attacks. Since these new techniques all have very low error rate, approximate attacks such as Double DIP and AppSAT have been proposed to find an almost correct key with low error rate. However, measuring the performance of an approximate attack is extremely challenging, since exact computation of the error rate is very expensive, while estimation based on random sampling has low confidence. In this paper, we develop a suite of scientific encryption benchmarks where a wide range of error rates are possible and the error rate can be found out by simple eyeballing. Then, we conduct a thorough comparative study on different approximate attacks, including AppSAT and Double DIP. The results show that approximate attacks are far away from closing the gap and more investigations are needed in this area.

I. Introduction

The growing participation of third party foundries in different stages of Integrated Circuit (IC) design from specification to physical implementation has given the semiconductor industry several emerging threats [2,4,6,11]. Among them, overproduction is the most serious one, since the precious design can be produced without notice of the designers [5]. Moreover, with the growing number of untrusted foundries, the possibility of Inside Foundry Attack (IFA) is also escalating. 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]. Several hardware protection techniques have been proposed to tackle the unauthorized production problem, of which logic encryption [3,8–10,12] attracts much attention.

In logic encryption, key-controlled gates are inserted 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 only if the key inputs have their predefined correct values. In this case, even though the inside foundry attacker is able to access the netlist and the external attacker can employ reverse engineering techniques to get the netlist from layout [16], they will not get functional circuit without knowing the correct key value. The inserted key-controlled gates can be further obfuscated to make it hard for attackers to directly remove them from the netlist [7].

However, the SAT-based attack [15] on logic encryption allows the attacker to narrow down the scope of correct key values utilizing an advanced satisfiability (SAT) solver. In order to defeat the SAT-based attack, different enhancing methods such as SARLock [18] and Anti-SAT [17] were proposed to make the time complexity of the SAT-based attack exponential. However, since these new techniques all have very low error rate, approximate attacks such as Double DIP [14] and AppSAT [13] have been proposed to find an approximate key which has low error rate with regard to the correct key.

An approximate attack is deemed to be more harmful than an exact attack. By deploying an approximate key in the circuit, an attacker can be assured to still make profit by selling the chip, since the tiny numbers of wrong outputs can almost not be discovered. However, these wrong outputs become a stealthy Trojan that could crash the whole system some day in the future. In other words, an approximation attack is more harmful since it can be viewed as an exact attack plus stealthy Trojan insertion.

A big challenge in studying approximate attacks are the measurement of the performance, which is given by the error rate of the returned key. On existing benchmarks, the error rate for a wrong key is either unknown or with a fixed tiny value (e.g. in SARLock or Anti-SAT). To find the exact error rate of a given key, every possible input must be simulated, which is exponentially expensive. To facilitate our comparative study, we have developed a suite of scientific benchmarks where the error rate of each key is different and can be adjusted while the error rate can be found by simple eyeballing of the key.

The contributions of this paper are as follows:

- We develop a suite of scientific encryption benchmarks where a wide range of error rate is possible, and the error rate can be exactly calculated from the key.
- We conduct a comparative study on existing approximate attacks. The results show that, the error rates achieved by these approximate attacks are similar to those by random key guess, and increasing the iteration number of approximate attacks does not decrease the error rates of the generated keys.

II. Preliminaries

In this section, we first overview the SAT-based attack [15] as an exact attack on logic encryption. Then, we discuss the Anti-SAT block [17] as an example of methods to defeat the original SAT-based attack. Finally, we review Double DIP [14] and AppSAT [13] as approximate attack algorithms.

A. SAT-based Attack

Using the SAT-based attack [15] on combinational logic encryption, almost all of the existing logic encryption methods [8, 9,12] can be successfully unlocked. The attack model assumes that the logic of the locked circuit is known, and the original circuit can be accessed as a black-box using an activated IC. The SAT-based attack is shown in Algorithm 1.

Supposing $C(X, K, Y)$ is the Conjunctive Normal Form (CNF) of the locked circuit with input X, key K, and output Y, the SAT-based attack iteratively finds the assignment to the CNF $C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge (Y_1 \neq Y_2)$ with constraints on the keys, until it is unsatisfiable. When X_i as an assignment of X is generated, its corresponding output Y_i from the original circuit is found, and these X_i and Y_i are used to 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 iteration will stop when the CNF is no longer

Algorithm 1 SAT-based attack

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

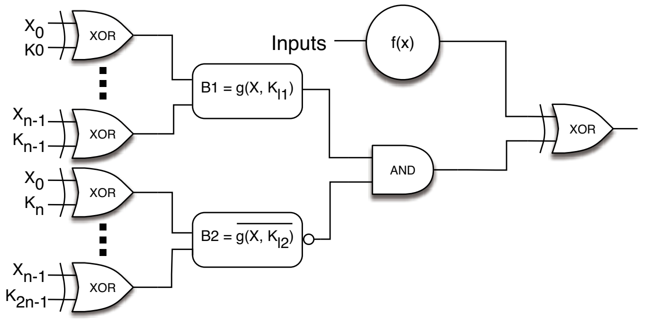

Fig. 1. The general design of Anti-SAT.

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 a SAT solver on the constraints. The SAT-based attack needs to use only a small number of X_i called Distinguishing Input Patterns (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.

B. Anti-SAT Block

One way to defeat the SAT-based attack is to increase the number of iterations in the process. The Anti-SAT block [17] shown in Fig. 1 is such a design. In Anti-SAT, a set of key-controlled gates (i.e. XORs) are inserted at the inputs of two logic blocks (i.e. B_1 and B_2), whose original functionalities are complementary. Therefore, $B_1 = g(X, K_{l1})$ and $B_2 = \overline{g(X, K_{l2})}$, where $|K_{l1}| = |K_{l2}| = n$. Hence, the key-size is $2n$. Then, the outputs of $B_1$and B_2 are fed into an AND gate, whose output is used to determine whether the output of the original circuit is flipped or not. In practice, one possible design of B_1 and B_2 is AND and NAND gates. Since the Anti-SAT block has $2n$ key bits, and the correct key happens when $K_0...K_{n-1} = K_n...K_{2n-1}$, there exists 2^n correct key combinations and $2^{2n} - 2^n$ wrong key combinations.

C. Double DIP

Double DIP [14] is an approximate decryption algorithm targeting SARLock [18]. The goal of Double DIP is to find the correct key of the traditional logic encryption technique K_1. Since wrong keys of SARLock will only cause very few wrong input-output pairs, Double DIP will terminate when K_1 is correct to avoid exponential iterations.

Double DIP shown in Algorithm 2 utilizes the following CNF with constraints to find two DIPs in each iteration:

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

If this CNF is satisfiable, X will prune out at least two wrong keys by adding the constraint of the original circuit evaluation.

Algorithm 2 Double DIP

```
Input: C and $eval$.
Output: $K^*$.
1: i = 1
2: $F_1 = C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge C(X, K_3, Y_1) \wedge C(X, K_4, Y_2)$
3: while $sat[F_i \wedge (Y_1 \neq Y_2) \wedge (K_1 \neq K_3) \wedge (K_2 \neq K_4)]$ do
4:     $X_i = sat\_assignment_X(F_i \wedge (Y_1 \neq Y_2) \wedge (K_1 \neq K_3) \wedge (K_2 \neq K_4))$
5:     $Y_i = eval(X_i)$
6:     $F_{i+1} = F_i \wedge C(X_i, K_1, Y_i) \wedge C(X_i, K_2, Y_i) \wedge C(X_i, K_3, Y_i) \wedge C(X_i, K_4, Y_i)$
7:     $i = i + 1$
8: end while
9: $K^* = sat\_assignment_{K1}(F_i)$
```

It can be proved that when applying Double DIP on SARLock, the traditional encryption part is guaranteed to be exactly decrypted when the algorithm terminates.

D. AppSAT

AppSAT [13] is another approximate attack algorithm based on random testing. Ideally, the error rate of the circuit with an approximate key should be below $\epsilon \in \mathcal{O}(\frac{1}{2^n})$, where n is the length of inputs. However, calculating ϵ precisely would require exponential queries itself, so heuristic methods for estimating the error is used for large functions [13].

The algorithm of AppSAT is as follows. First, it uses SAT-based attack to prune out key values with a certain number of DIPs; then, a SAT solver is utilized to provide a key value satisfying all these DIPs. To evaluate the correctness of the key value, random testing is adopted to estimate the error rate of the key. If the estimated error rate is below ϵ, the key value is considered as an approximate key with ϵ error rate. Otherwise, the random sampling that resulted in a disagreement will be added to the SAT formula as a new constraint. The combination of the SAT-based attack and random testing is repeated until the estimated ϵ is below the threshold.

III. Challenges for Approximate Attacks

An approximate attack on logic encryption is more harmful than an exact attack, since it can be viewed as an exact attack combined with a stealthy Trojan insertion. Thus, the investigation of approximate attacks and their prevention has become a critical research topic in logic encryption.

Here, we are trying to understand how effective the existing approximate attacks are. It has been shown in [13] that AppSAT can effectively find the correct keys of the traditional encryptions that are mixed with Anti-SAT for almost all the benchmarks. To explain why AppSAT is effective, they also plotted the error rates after each iteration in the SAT-based attack, and demonstrated that, with 10 random samples being reinforced in each iteration, the error rate decreases much faster.

However, we want to point out that *only measuring the simulated attack time on logic encryption is misleading since each query of the correct circuit will be more expensive in the real attack.* Remember that each query of the correct circuit needs to scan in the input, proceed in one clock cycle, and then scan out the output. When 10 random samples are taken in each iteration of the SAT-based attack, it simply means that 11 times of queries are needed.

Therefore, we just want to conduct the following experiments. On each benchmark with traditional encryption mixed with Anti-SAT, we first run AppSAT with the same parameters as in [13] and record the total number of queries on the correct circuit. Then, we run the original SAT-based attack with the same number of queries. The experimental result is shown in Table I. Out of 36 benchmarks, 27 and 29 benchmarks can be successfully decrypted by the SAT-based attack and AppSAT respectively. It indicates that these two attacks performs almost the same and we would better have further investigation.

TABLE I
APPSAT AND THE SAT-BASED ATTACK WITH THE SAME QUERY NUMBER ON BENCHMARKS ENCRYPTED WITH ANTI-SAT + A TRADITIONAL ENCRYPTION METHOD [12].

	AppSAT		SAT-based attack	
overhead	5%	10%	5%	10%
apex2	no	no	no	no
apex4	yes	no	yes	no
c1355	yes	yes	yes	yes
c1908	yes	yes	yes	no
c3540	yes	yes	yes	yes
c432	yes	yes	yes	yes
c499	yes	yes	yes	yes
c5315	yes	yes	yes	yes
c880	yes	yes	yes	no
dalu	yes	yes	yes	yes
ex1010	no	no	yes	no
ex5	yes	yes	yes	yes
i4	yes	yes	yes	yes
i7	yes	yes	yes	yes
i8	yes	yes	yes	yes
i9	yes	yes	yes	yes
k2	yes	yes	no	yes
seq	no	no	no	no

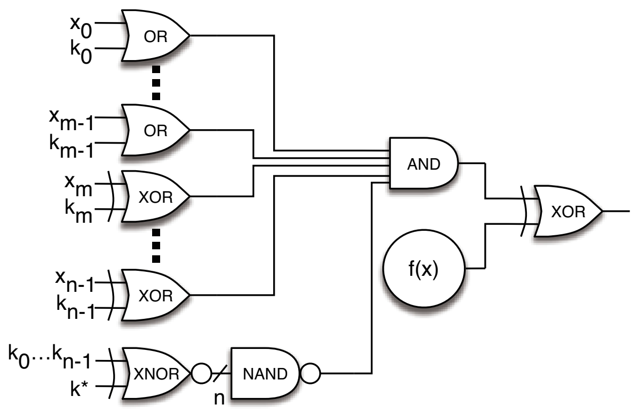

Fig. 2. The general design of ECE as a suite of scientific benchmarks.

However, one big challenge for studying approximate attack methods is the lack of scientific measure of their performances. Different from exact attacks, where the correctness can be easily measured by comparing the keys or by comparing the circuits if there are more than one correct keys, the performances of approximate attacks cannot be easily measured by the generated keys or even the generated circuits. For two approximate attacks, one is better than the other if the circuit generated by one has less error rate than the other. Exact measure of error rate needs to do circuit simulation for all possible inputs or to do SAT queries to find all the errors one by one. None of them is cheap.

Not any better is the current practice of using the combination of a traditional encryption and a specific encryption against the SAT-based attack such as Anti-SAT. The easy thing to report is the number of benchmarks where the key of the traditional encryption is correct. However, if an approximate attack could not get the key of the traditional encryption correct, which is very common for large or complex benchmarks, we get lost again. Measuring how many bits are correct in the key of the traditional encryption is of no use, since a mistake on one bit may have more errors than a mistake on many other bits. Using random sampling for error rate measurement is relatively cheap but the accuracy is highly in doubt.

For these reasons, it is critical for the investigation of approximate attacks to develop a set of scientific benchmarks to measure their performance. The desired properties for the scientific benchmarks include:

1. Different keys have different error rates;
2. The error rate is known for each key;
3. The error rate is adjustable;
4. The benchmarks are difficult to be decrypted by the SAT-based attack.

IV. ERROR-CONTROLABLE ENCRYPTION

In this section, we present a design of such a suite of scientific benchmarks, named *Error-Controlable Encryption (ECE).* The general design of ECE is given in Fig. 2.

Here we assume that the original circuit $f(x)$ has n input bits and the logic encryption has n key bits. We are going to select a correct key K^*, and a number $m < n$ as a design parameter. As shown in the figure, we are going to have OR gates of x_i and k_i for all $i \in 0..m-1$ and to have XOR gates of x_i and k_i for all $i \in m..n-1$. Then, we feed outputs of all these gates to an AND gate. $k_0...k_{n-1}$ is compared with correct key value K^*, and a signal will be generated by XNOR and NAND gates to the same AND gate. The output of the AND gate is used to flip the output of $f(x)$ by an XOR gate. Here, we do not worry about structural attacks to these benchmarks, since we can do resynthesis to obfuscate the circuit structure.

The ECE scientific benchmarks guarantee that the error rate for a key can be exactly calculated. If a key is correct, then $k_0...k_{n-1}$ is equal to correct key value K^*, and a zero signal will be generated by XNOR and NAND gates. It forces the output of the AND gate to be zero so that the flipping is disabled. Otherwise, $k_0...k_{n-1}$ is a wrong key, and the NAND gate outputs one. The error rate depends on how many key bits connecting to OR gates are equal to one. Flipping signal happens to be one only when outputs of all XOR and OR gates become one. All XOR gates output one when $x_i = \overline{k_i}$ for all $i \in m..n-1$. To make all OR gates output one, for i_{th} OR gate, $0 \leq i \leq m-1$, x_i can be either zero or one if $k_i = 1$, and x_i has to be one if $k_i = 0$.

Assume for $0 \leq i \leq m-1$, the number of $k_i s$ so that $k_i = 1$, is equal to l. Then, there exists 2^l possible inputs which enable the flipping signal, and the error rate of the wrong key is 2^l / 2^n = 2^{l-n}. Now we can prove the following theorem for the ECE scientific benchmarks.

Theorem 1. *The ECE scientific benchmarks will have different error rates ranging from 2^{-n} to 2^{m-n} for a wrong key. The error rate is known for each key, and the minimal number of iterations for the SAT-based attack is 2^{n-m}.*

PROOF. The upper bound of the error rate happens when k_i for all $i \in 0..m-1$ is set to one. In that case, the value of the flipping signal depends on the result of XOR gates. For a random assignment of k_i for all $i \in m..n-1$, the flipping signal is 1 only if $x_i = \overline{k_i}$ for all $i \in m..n-1$. So the error rate is $2^m / 2^n = 2^{m-n}$.

The lower bound of the error rate happens when k_i for all $i \in 0..m-1$ is set to zero. Then, the flipping signal is one only if $x_i = 1$ for $i \in 0..m-1$ and $x_i = \overline{k_i}$ for $i \in m..n-1$. In that case, the error rate is 2^{-n}.

Any other key values will have the error rate ranging from 2^{-n} to 2^{m-n}. To solve the correct key K^*, the SAT-based attack should prune out all wrong keys. Since for an assignment of X, only keys with $k_i = \overline{x_i}$ for all $i \in m..n-1$ are possible to be pruned in each iteration, and there exists 2^{n-m} combinations for $x_m...x_{n-1}$, therefore the number of iterations for the SAT-based attack is at least 2^{n-m}. □

The analysis of the ECE scientific benchmarks illustrates they are not only a suite of benchmarks to accurately measure the performance of approximate attacks, but also a robust logic encryption technique to defeat the SAT-based attack. By assigning different values to m, the error rate of a key and the necessary number of iterations for the SAT-based attack to decrypt the correct key are adjustable. An interesting trade-off is that increasing m increases the upper bound of the error rates, but also decreases the number of iterations for the SAT-based attack. Designers can also introduce randomness to the benchmarks by randomly selecting K^*, randomly selecting m indices for the OR gates (the remaining will be XOR gates), and randomly inserting inverters after each key bit to further obfuscate the circuit.

TABLE II
ITERATIONS FOR APPROXIMATE TECHNIQUES TO ATTACK ECE SCIENTIFIC BENCHMARKS.

	Iterations			
	SAT	AppSAT	RS Attack	Double DIP
apex2	500	24	1	500
apex4	223	24	1	164
c1355	500	24	1	500
c1908	500	24	1	458
c3540	500	24	1	500
c432	500	24	1	500
c499	500	50	1	500
c5315	500	24	1	500
c880	500	24	1	500
dalu	500	24	1	500
des	500	24	4	500
ex1010	500	157	5	211
i4	500	24	1	500
i7	500	24	1	500
i8	500	24	1	500
i9	429	24	1	500
k2	500	24	1	500
seq	500	24	1	500

V. EXPERIMENTAL RESULTS

We perform four advanced logic decryption techniques on ECE scientific benchmarks, and compare the exact error rate of approximate keys with the error rate of a random generated key. The result shows that existing approximate attack techniques still need to be further developed. It also shows that, the error rate generated by these approximate attacks is similar to the error rate of a random generated key, and the generated error rate does not decrease in each iteration.

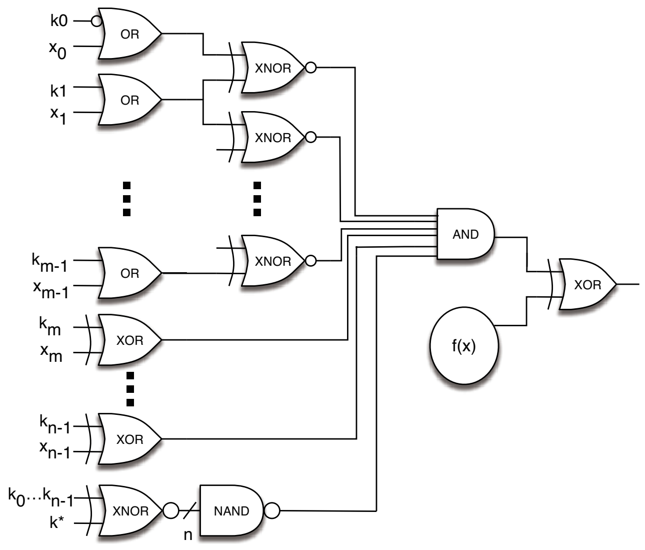

Fig. 3. ECE benchmarks with extra XNOR gates and inverters.

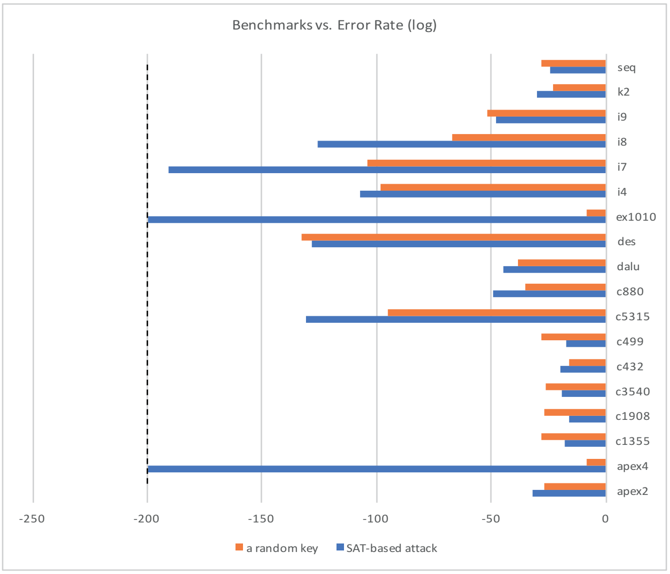

Fig. 4. A comparison of the error rate between a key solved by the SAT-based attack and a random key.

One potential issue for ECE is that a SAT solver may start to try easy assignments such as all zeros or all ones. To avoid the SAT solver assigns all zeros to k_is for all $i \in 0..m-1$ to have a small error rate, we invert half of the key inputs from k_0 to k_{m-1}, and add extra XNOR gates as equivalence checking shown in Fig. 3. We assign the number of XOR gates to seven, so that the SAT-based attack requires at least 2^7 iterations to prune out wrong keys.

We apply four attack techniques on ECE, which are the SAT-based attack, AppSAT, Double DIP, and Random Sampling (RS) attack. RS attack randomly guesses a key and uses random samples to evaluate the error rate of the guessed key. If the error rate exceeds the threshold, the random samples that result in disagreement compared with the original circuit will be added into the SAT solver to further constrain the key. For AppSAT and RS attack, we query 200 random samples five times to estimate the error rate and collect random samples with disagreeing results. If there is no random sample which results in a different output for five times, the algorithm will be terminated and a key will be generated by the SAT solver. For AppSAT we run the random sampling after every 24 iterations of the SAT-based attack, and since the SAT-based attack and Double DIP may trap into exponential iterations, we set the number of maximum iterations to 500 as the condition of termination.

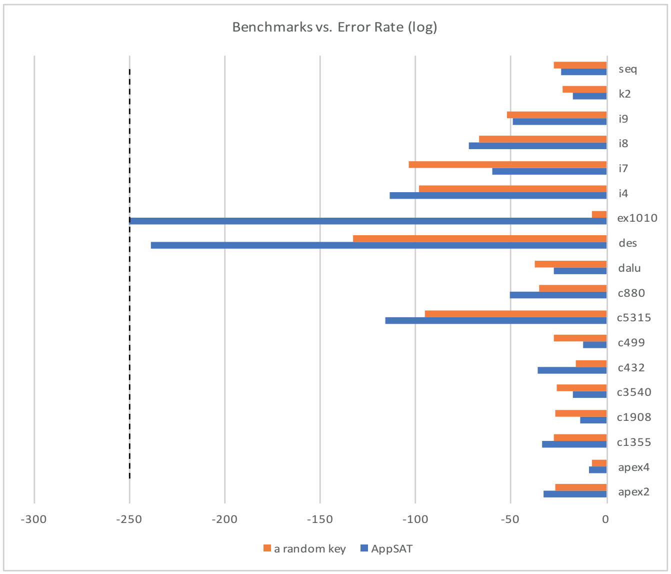

Fig. 5. A comparison of the error rate between a key solved by AppSAT and a random key.

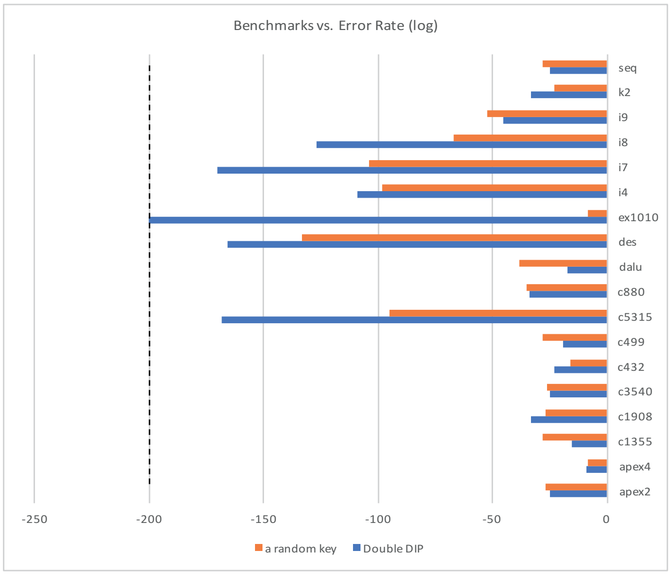

Fig. 6. A comparison of the error rate between a key solved by Double DIP and a random key.

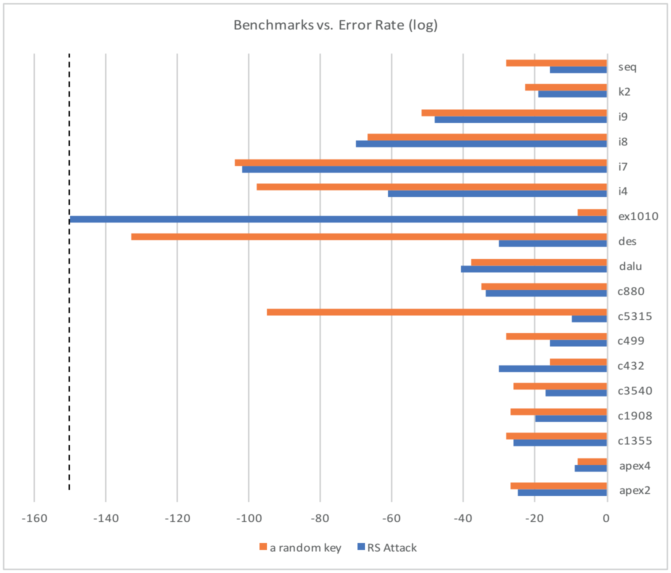

Fig. 7. A comparison of the error rate between a key solved by RS attack and a random key.

Table II demonstrates the number of iterations for these approximate methods to attack ECE. The queries that are required for random sampling in AppSAT and RS attack are not shown in the table. An interesting result is that for RS attack, since initially there is no DIP in the SAT solver, a random key is generated, and it already satisfies the requirement of error rate by the random sampling.

Fig. 4, 5, 6, 7 compare the error rate of keys from different approximate attacks with the error rate of a random generated key. The dashed line indicates the correct key is solved so that the error rate is $2^{-\infty}$. Two benchmarks apex4 and ex1010 are possible to be successfully attacked, since for these two benchmarks the length of inputs are limited (i.e. 10 bits). For other complex benchmarks, correct keys cannot be solved. The comparison illustrates the error rate of a random generated key is at the same level compared with the error rate of the returned keys from other attack techniques, which indicates these approximate decryption techniques are no better than random guessing while attacking ECE.

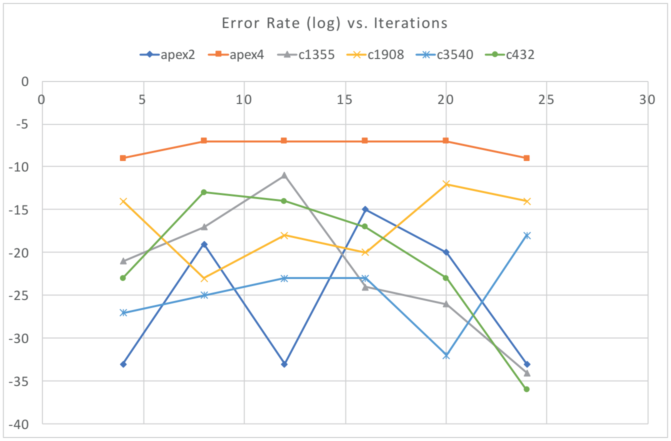

Fig. 8. The error rate of temporary keys generated by AppSAT with increasing of iterations.

To study whether the error rate of the returned keys is related to the number of iterations of the logic decryption techniques,

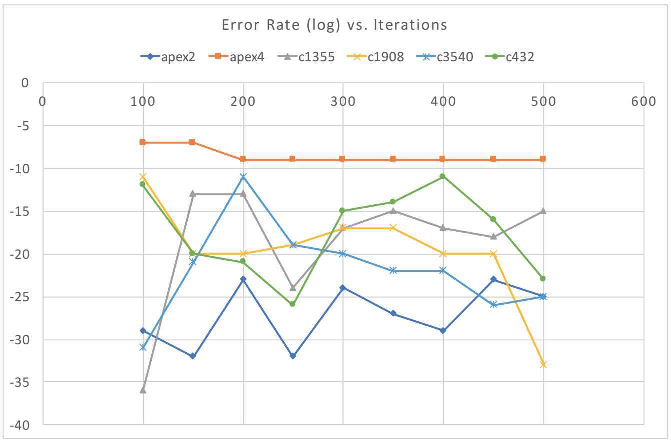

Fig. 9. The error rate of temporary keys generated by Double DIP with increasing of iterations.

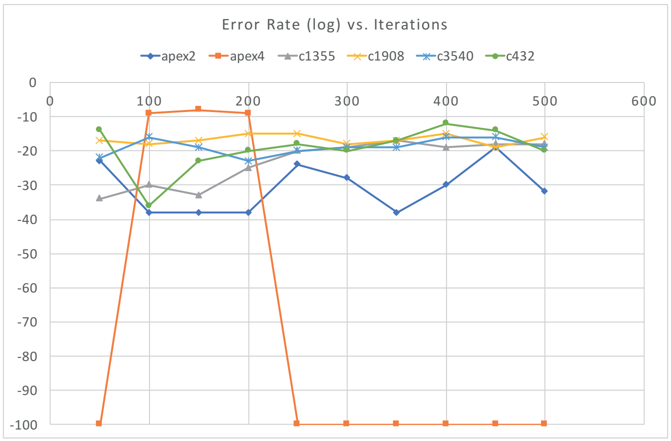

Fig. 10. The error rate of temporary keys generated by the SAT-based attack with increasing of iterations.

we use the SAT solver to generate a set of temporary keys during the process of the decryption and explore if more iterations are beneficial to decrease the error rate of the keys. Fig. 8, 9 and 10 show the error rate of the temporary keys with increasing the number of iterations. Even though more DIPs are put into the SAT solver, the error rate does not have a meaningful trend to decrease and appears to be random. Thus, more advanced approximate logic decryption techniques should be investigated.

VI. Conclusion and Acknowledgement

In this paper, we proposed a suite of scientific benchmarks called Error-Controllable Encryption (ECE) which has different error rates for different keys, and the error rate can be easily calculated for each key. We performed four existing approximate logic attacks on ECE scientific benchmarks and found that they are not better than random key guessing. Increasing the iteration number of these attacks does not improve the error rate of the generated keys either. Therefore, approximate attacks are far away from being effective and further investigations are needed.

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

VII. References

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

[2] Miron Abramovici and Paul Bradley. Integrated circuit security: new threats and solutions. In *Workshop on Cyber Security and Information Intelligence Research*, page 55, 2009.

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

[4] Swarup Bhunia, Michael S Hsiao, Mainak Banga, and Seetharam Narasimhan. Hardware trojan attacks: threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, 2014.

[5] Ujjwal Guin, Domenic Forte, and Mohammad Tehranipoor. Anti-counterfeit techniques: from design to resign. In *14th International Workshop on Microprocessor Test and Verification*, pages 89–94, 2013.

[6] Ryan Kastner and Ted Huffmire. Threats and challenges in reconfigurable hardware security. Technical report, UCSD, ECE, 2008.

[7] Li Li and Hai Zhou. Structural transformation for best-possible obfuscation of sequential circuits. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*, pages 55–60, 2013.

[8] Jeyavijayan Rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. Logic encryption: A fault analysis perspective. In *Proc. DATE: Design Automation and Test in Europe*, pages 953–958, 2012.

[9] Jeyavijayan Rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of logic obfuscation. In *Proc. of the Design Automation Conf.*, pages 83–89, 2012.

[10] 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.

[11] Masoud Rostami, Farinaz Koushanfar, Jeyavijayan Rajendran, and Ramesh Karri. Hardware security: Threat models and metrics. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 819–823, 2013.

[12] Jarrod A. Roy, Farinaz Koushanfar, and Igor L. Markov. EPIC: Ending piracy of integrated circuits. In *Proc. DATE: Design Automation and Test in Europe*, 2008.

[13] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. AppSAT: Approximately deobfuscating integrated circuits. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*, pages 95–100, 2017.

[14] Yuanqi Shen and Hai Zhou. Double dip: Re-evaluating security of logic encryption algorithms. In *Proc. ACM Great Lakes Symposium on VLSI*, pages 179–184, 2017.

[15] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*, pages 137–143, 2015.

[16] Randy Torrance and Dick James. The state-of-the-art in ic reverse engineering. In *Conference on Cryptographic Hardware and Embedded Systems*, pages 363–381. 2009.

[17] Yang Xie and Ankur Srivastava. Mitigating SAT attack on logic locking. In *Conference on Cryptographic Hardware and Embedded Systems*, pages 127–146, 2016.

[18] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan J V Rajendran, and Ozgur Sinanoglu. SARLock: SAT attack resistant logic locking. In *Proc. IEEE International Symposium on Hardware Oriented Security and Trust*, pages 236–241, 2016.