Deep RNN-Oriented Paradigm Shift through BOCANet: Broken Obfuscated Circuit Attack

Fatemeh Tehranipoor*, Nima Karimian**, Mehran Mozaffari Kermani†, and Hamid Mahmoodi*
* School of Engineering, San Francisco State University, USA
** Electrical and Computer Engineering, University of Connecticut, USA
† Computer Science and Engineering Department, University of South Florida
[tehranipoor@sfsu.edu]

***Abstract*—This is the first work augmenting hardware attacks mounted on obfuscated circuits by incorporating deep recurrent neural network ($D-RNN$). Logic encryption obfuscation has been used for thwarting counterfeiting, overproduction, and reverse engineering but vulnerable to attacks. There have been efficient schemes, e.g., $satisfiability-checking$ (SAT) based attack, which can potentially compromise hardware obfuscation circuits. Nevertheless, not only there exist countermeasures against such attacks in the state-of-the-art (including the recent delay+logic locking (DLL) scheme in $DAC'17$), but the sheer amount of time/resources to mount the attack could hinder its efficacy. In this paper, we propose a deep RNN-oriented approach, called $BOCANet$, to (i) compromise the obfuscated hardware at least an order-of magnitude more efficiently ($>20X$ faster with relatively high success rate) compared to existing attacks; (ii) attack such locked hardware even when the resources to the attacker are only limited to insignificant number of I/O pairs ($<$ 0.5%) to reconstruct the secret key; and (iii) break a number of experimented benchmarks ($ISCAS-85$ $c423$, $c1355$, $c1908$, and $c7552$) successfully.**

***Index terms*—** Deep recurrent neural network (D-RNN), hardware obfuscation, logic encryption.

I. INTRODUCTION

Over the past two decades, many of the leading semiconductor companies have become fabless due to the fact that the increasing costs and complexity confine them not to design, test, fabricate, and package ICs. The trend in recent past has been to move towards the globalization of supply chains, and, unfortunately, there exist a number of security threats associated with such exposure. It is well-known that various vulnerabilities introduce numerous opportunities for malicious parties to engage in IP piracy, counterfeiting, and reverse engineering. Hardware obfuscation is a state-of-the-art technique that can be utilized to protect semiconductor IPs at various levels of abstraction [1].

Hardware obfuscation technique is the process of transforming a function F to another function $F(O)$, which has the same functionality as F, e.g., input/output nature, and conceals (locks) the functionality of F (as shown in Fig. 1(a)) and/or the structure of an IC from attackers. Through logic encryption, a variant of hardware obfuscation techniques, a given combinational circuit is modified with newly-added inputs, denoted as key inputs, to ensure that the encrypted/locked circuit operates correctly, i.e., produces correct outputs, under a specific key value assertion (valid key). In logic encryption obfuscation, the correct key must not be accessible to the untrusted foundry. Otherwise, the foundry can use it to activate overproduction of ICs and sell them improperly and illegally [2]. Unfortunately, logic encryption has been recently proven to be compromised by a number of research groups, leaving it potentially vulnerable to various attacks in order to learn the correct key (secret key). Examples of such attacks include satisfiability-checking (SAT) based attack in [3] proposed recently by Subramayan *et al.*, a fault analysis attack that directly propagates the key bits to the circuit outputs proposed in [4], and a randomized, local key-searching algorithm to search the key that can satisfy a subset of correct input/output patterns proposed in [5]. Recent research has also focused on attacks/countermeasures for logic locking [6]–[8]. Nevertheless, we note that although SAT attack can successfully break several existing logic locking techniques, not only there exist countermeasures against such attacks in the state-of-the-art (including the recent delay+logic locking (DLL) scheme in [2]), but the sheer amount of time/resources to mount the attack could hinder its efficacy. Note that SAT attack takes few hours to effectively predict the secret key.

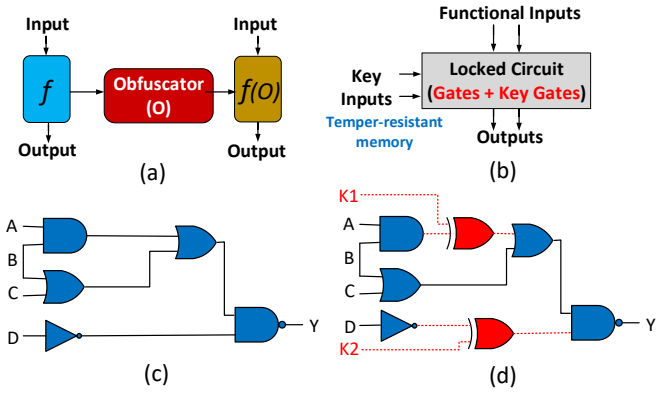


Figure 1. Schematic of (a) a general obfuscation scheme, (b) a general logic locking obfuscation, (c) an original circuit without key gates and key inputs, and (d) an encrypted circuit with key gates and key inputs.

In this paper, we propose a new attack model against logic encryption obfuscation using deep recurrent neural network (D-RNN) denoted as BOCANet: Broken obfuscated circuit attack (see [9] for background). Given a very small number of input/output patterns from the logic encryption obfuscated circuit, this attack enables not only to reconstruct the secret key but also predict the unknown outputs by giving the input patterns of the circuit (without the need of secret key) and vice versa. Based on our results, BOCANet can effectively break logic encryption circuits in order to reconstruct the secret key, achieving relatively high success rate. To the best of our knowledge, this is the first work augmenting hardware attacks mounted on hardware obfuscated circuits using deep learning technique.

Specifically, the major contributions of the proposed work are as follows:

- *Broken obfuscated circuit attack, BOCANet*: We introduce BOCANet, a novel attack framework scheme. BOCANet incorporates D-RNN to compromise the secret key from obfuscated hardware at least an order-of-magnitude more efficiently ($> 20X$ faster with relatively high success rate) compared to SAT attacks.
- *BOCANet strength properties*: In order to objectively measure the success of the proposed attack, we test our BOCANet scheme to break a number of experimented benchmarks (ISCAS-85 c423, c1355, c1908, and c7552) successfully with 32-bit key size (100% success rate), 64-bit key size (94% success rate), 128-bit key size (92% success rate), and 256-bit key size (89% success rate), respectively.
- *BOCANet obliviousness of secret key*: Through our experiments, we show that BOCANet is oblivious of the secret key such that given a number of trained input/output patterns to D-RNN, it is capable of predicting any unknown I/O pairs without the knowledge of circuit's secret key.

We first present the preliminaries in Section 2. In Section 3, we present the proposed scheme, BOCANet. Section 4 presents the results of our proposed attacks assessments on logic locking obfuscated circuits along with evaluating four ISCAS-85 benchmarks. We conclude the paper in Section 5.

II. PRELIMINARIES

A. Logic-Based Hardware Obfuscation

Logic-based hardware obfuscation (or logic encryption, logic locking) is an emerging and promising technique to thwart the threat of counterfeiting, overproduction, and reverse engineering by an untrusted foundry [10]–[12]. Through logic locking, a given combinational circuit is modified with newly-added inputs to ensure that the encryption circuit produces correct outputs under a specific key value assertion (valid key). Upon applying a wrong key, the encrypted design will exhibit a wrong functionality, i.e., produces wrong outputs. Since the design is encrypted by the designer, the foundry cannot use any copies or overproduce ICs without the secret keys. Logic encryption hides the functionality and the implementation of a design by inserting additional gates into the original design. We note that the gates inserted for encryption are key-controlled gates. Fig. 1(b) shows a general schematic of logic locking obfuscation technique which contains functional input (input patterns to the circuit), key input (secret key that is stored in tamper resistant memory), locked circuit (including normal gates and added gates for key gates), and outputs from the locked circuit. Fig. 1(c) and Fig. 1(d) illustrate a circuit without/with key gates, respectively (modulo-2 addition through XORing with K1 and K2).

B. SAT Attacks

Satisfiability-checking (SAT) based attack is a newly-proposed attack [3], threatening the security of logic encryption circuits by deciphering a functionality-correct key (w.r.t. combinational logic) of most locking techniques within a few hours [2]. The main step in the SAT-based attack is to use two copies of the encryption logic circuit with the same input but different keys (lock-keys), under a given constraint, to check whether it is still possible to generate different outputs from the locked circuit. Such small number of input/output pairs are denoted as differentiating input patterns (DIPs) that can be used to eliminate incorrectly-guessed keys. Each DIP is then used to query the original circuit blackbox to get the correct output. The DIP with output is then used to further constrain the keys under consideration. In fact, the idea of using DIP is to exclude at least one wrong key from consideration. Finding DIPs requires a sequence of SAT formula that can be solved by SAT solvers. In order to defeat SAT attacks, a number of research works have proposed inserting additional SAT attack resistant logic blocks, e.g., the Anti-SAT block by Xie *et al.* [13] and SARLock by Yasin *et al.* [14].

AppSAT is another attack to logic locking which has been presented in HOST 2017. AppSAT [15] is a technique for approximate deobfuscation based on the active learning (semi-supervised machine learning) with random querying and intermediate error estimation. The main goal in active-learning is to find a querying strategy that minimizing the number of queries required to learn the target function. While active learning is sample-efficient, it can be computationally expensive since it requires iterative retraining. To speed this up, we propose a lightweight architecture based on LSTM. Our model is not only computationally much more efficient than AppSAT, but also its accuracy is higher than that of AppSAT mechanism can provide. Another Advantage of our work compare with other attacks is that when a different key is applied to the same benchmark, deep learning does not need to be trained from scratch one more time. Consequently, we can apply transfer learning by taking a fully-trained model for the existing BOCANet and retrain from the existing weights for new benchmark to deobfuscate less than a minutes. The memory usage is remained below 1GB with a quarter of our CPU usage.

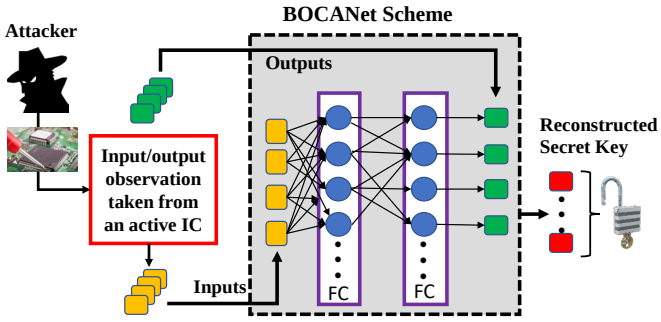


Figure 2. Block diagram of our proposed BOCANet scheme.

III. OVERVIEW OF THE PROPOSED BOCANET SCHEME

A. *Data Acquisition*

We consider open source obfuscation benchmarks [16] for our deep learning-based reverse engineering experiments. These benchmarks are combinational logic circuits with complexities ranging from less than a thousand to more than five thousands gates. These benchmarks utilize the logic-based obfuscation technique using random key gate insertion with key sizes ranging from 32 bits to 256 bits. We note that these keys are typically programmed in an on-chip non-volatile memory in a trusted environment after manufacturing. The attack model assumes that there is access to a fabricated chip with programmed keys. The size of the key is known information as it is visible form the circuit interface, but the key value is unknown. With access to a chip with programmed key, an electrical test can be performed to extract an input/output stimulus/response table. The unknown part is the secret key, and the goal of the deep learning algorithm is to find the key from a given input/output stimulus/response table.

To generate input/output stimulus/response table, we setup a SystemVerilog testbench [17] for each benchmark to simulate its gate level netlist with randomly-generated input stimuli and obtain corresponding output responses. For large number of inputs, it would be prohibitive to simulate all possible input combinations as we would run out of either time and/or computer memory space to generate and store all combinations. Hence, we generate as many random stimuli as sufficient for the deep learning algorithm to come to a reasonable estimate of the key. The accuracy of the key estimate is improved by providing more input/output stimulus/response data. For the purpose of this experiment, we manually set a key for a design under test, and check to see if the deep learning algorithm can determine the key. We would like to emphasize that this does not confine the proposed scheme, and the proposed approach in this paper is oblivious of the chosen key.

B. *Attack Scheme*

In this work, we introduce a new attack called BOCANet attack in which we use deep learning through recurrent neural network, i.e., D-RNN, to predict the secret key. The D-RNN is a deep learning-based algorithm that can perform the same task for every element of a sequence (data), with the output being

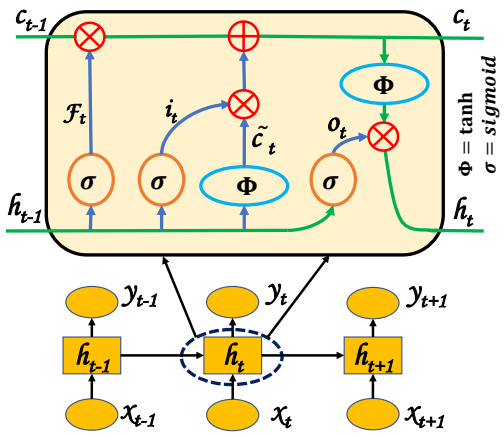


Figure 3. Block diagram of deep RNN with long short-term memory (LSTM).

dependent on the previous computations. In the BOCANet attack, our assumption is that there exist some input/output patterns for an attacker (from an obfuscated circuit) in order to predict the secret key. For these experiments, we have considered four benchmarks [16] with different key sizes (32-bit, 64-bit, 128-bit, and 256-bit). Here, we also assume that an attacker has access to only less than 0.5% of the total I/O pairs available from the obfuscated circuit. Fig. 2 illustrates our proposed BOCANet attack model to obfuscated circuits using deep learning techniques. As shown in this figure, an attacker may have access to a set of I/O pairs of an obfuscated circuit by probing (or somehow accessing) them. By knowing the correct I/O pairs, deep learning is able to train itself in order to predict a secret key that has been implemented in the obfuscated hardware. In order to train our BOCANet scheme, we have implemented deep RNN [18] with long short-term memory (LSTM) [19]. More details on our BOCANet approach is discussed in the following.

C. *Deep RNN Implementation*

Recurrent neural networks have recently attracted prominent attention for modeling variable length sequences [20]. A D-RNN is a deep structure of a neural network that operates in time. At each time step, it accepts an input vector, updates its hidden layer through non-linear activation functions, and uses it to predict its output. Since D-RNN's hidden layer can store high-dimensional information, and its nonlinear dynamics can implement a powerful computation, it forms D-RNN to form a strong model to perform modeling and prediction functions with a high complex structure. In this paper, we explore deep extension of basic RNN to model our BOCANet for reconstructing secret key from obfuscated circuits. Given a sequence of inputs $x_1, x_2, x_3, ..., x_p$ and outputs $y_1, y_2, y_3, ..., y_q$, we want to predict secret-key $k_1, k_2, x_3, ..., k_n$. In order to do that, we train I/O pairs with LSTM RNN architecture.

LSTM unit refers to a specific architecture of RNNs, introduced by Hochreiter & Schmidhuber [21], targeting to tackle long-term dependencies challenge unsolved in earlier RNN architectures. When learning time-series data, RNNs aim to learn the patterns repeatedly happened in the past by sharing

the states that are decomposed into multiple layers in order to gain properties from 'deep' architectures [19]. Fig. 3 presents a D-RNN with LSTM cell architecture. LSTM cells have a special sharing parameter vector called memory parameter vector c_t deployed to keep the memorized data. In each time stage, the memory parameter has three operations: (i) Discarding useless data from memory vector c_t, (ii) adding new data i_t selected from input x_t and previous sharing parameter vector h_{t-1} into memory vector c_t, and (iii) deciding new sharing parameter vector h_t from memory vector c_t. As shown in the LSTM cell, the sharing memory parameters, h_t, are passing through various time stages only with two operations to *memorize* new data and *forget* time-out memories. Thus, the sharing memory can conduct useful data for an adequately-long time and results in enhancing the RNN performance:

$$\begin{cases} f_t \leftarrow \sigma_g(W_{xf}[h_{t-1},x_t]+V_{cf}c_{t-1}+b_f) \\ i_t \leftarrow \sigma_g(W_{xi}[h_{t-1},x_t]+V_{ci}c_{t-1}+b_i) \\ o_t \leftarrow \sigma_g(W_{xo}[h_{t-1},x_t]+V_{co}c_{t-1}+b_o) \\ c_t \leftarrow f_t * c_{t-1}+i_t * tanh(W_{xc}[h_{t-1},x_t]+b_c) \\ h_t \leftarrow o_t * tanh(c_t) \end{cases} \quad (1)$$

where f_t, i_t, o_t, c_t, and h_t are the forget gate, input gate, output gate, cell, and hidden state output, respectively.

The backpropagation algorithm is used to train the LSTM net, i.e., weights are updated based on the gradient descent error of the output. According to the gradient descent rule, the weights are modified towards minimizing the square error of the output:

$$E \leftarrow \sum_{q=1}^{P}(D_q - Z_q)^2 + E \quad (2)$$

where Z_q is the predicted output of the q-th residue in the training dataset, D_q is the target output vector, and p is the number of training patterns. The learning procedure is carried out by updating the weights W proportional to the gradient descent of the error for every training pattern as follows:

$$\delta_{zq} : (d_q - z_q)z_q(1-z_q) \qquad q = 1,2,...,P \quad (3)$$

$$\delta_{yj} : y_j(1-y_j)\sum_{q=1}^{P}\delta_{zq}W_{iq} \qquad j = 1,2,...,n \quad (4)$$

$$\begin{cases} \Delta W_{jq} \leftarrow \eta y_j \delta_{zq} + \alpha \Delta W_{jq} \\ W_{jq} \leftarrow W_{jq} + \Delta W_{jq} \end{cases} \quad (5)$$

where η is called the constant learning rate. To sustain the effect of the past input patterns, the momentum term α is always appended and the weights are optimized through (5). The error vector δ_{zq} is propagated to the back layers and the weights are moderated based on the back propagated error. The constant learning rate and the momentum term are adjusted to 0.01 and 0.9, respectively. The network weights are initialized with small random values within $[-0.05, 0.05]$ interval.

Fig. 4 demonstrates the training process of experimented ISCAS-85 benchmarks in D-RNN in order to reconstruct the secret key. As shown in the figure, there are numerous number

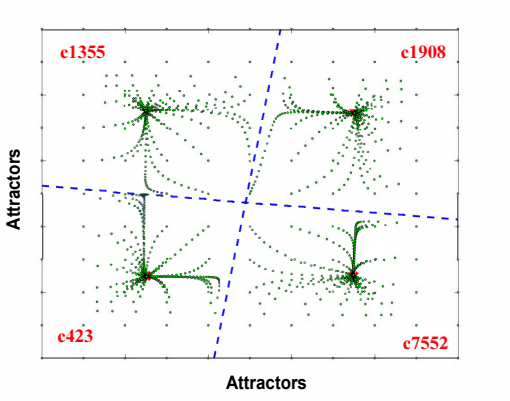
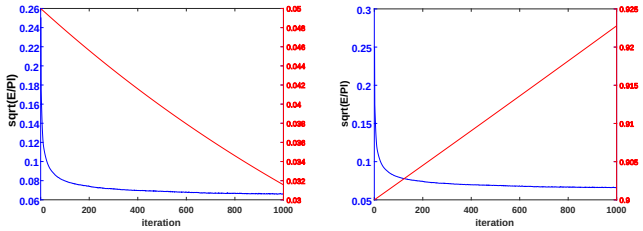


Figure 4. Neuron attractors under 4 different benchmark secret keys.



(a) (b)

Figure 5. The MSE error (blue curve) *vs.* (a) training step (red line) and (b) momentum (red line).

of green circle points (nodes) that indicate neuron attractors. These nodes try to settle to a stable pattern (here referred to as secret key) which is shown in the figure by 4 red cross points (×). More precisely, these attractors network are a set of N network nodes from each benchmark connected in such a way that their global dynamics become stable to predict the secret key from the obfuscated circuits. Furthermore, in order to evaluate the performance of our attack model (BOCANet), one potential metric is mean-square-error (MSE). MSE is a cost function that measures the average of the squares of the errors; the difference between the estimator, i.e., obfuscated circuit output, and what is estimated, i.e., predicted output. Fig. 5(a) illustrates the impact of training step *vs.* momentum coefficient on MSE. As can be seen in Fig. 5(a), by decreasing the training step in each epoch, the D-RNN weight can be adjusted more efficiently than constant training. Fig. 5(b) represents how an increase in momentum has impact on the error rate of trained weights.

IV. EXPERIMENTAL RESULTS

To break logic encryption obfuscated implementation, we have conducted 3 different DL-based attacks: i) Attack1-Key Reconstruction: In this scenario, the input/output pairs collected from several ISCAS-85 benchmarks from [16] are available and the aim is to reconstruct the secret key; ii) Attack2-Output Guessing: In this attack, the attacker applies a set of inputs to deep learning in order to figure out the corresponding outputs (unknown outputs); and iii) Attack3-

Table I
EXPERIMENTED OBFUSCATED ISCAS-85 BENCHMARKS, THEIR CHARACTERISTICS, AND THE REQUIRED COMPROMISE TIME

	# Inputs	# Outputs	Key-size	Attack Time
c423	32	7	32 Bits	**6 mins**
c1355	41	32	64 Bits	**11 mins**
c1908	33	25	128 Bits	**19 mins**
c7552	207	108	256 Bits	**35 mins**

Input Guessing: In this attack model, the attacker tries to guess the input patterns (unknown inputs) while the outputs are given to the deep learning infrastructure. As indicated earlier, Attack2 and Attack3 are independent from secret key. In other words, they do not require key for prediction of I/Os. We assume that the attacker has access to a set of I/O pairs and the key is unknown to him/her.

A. *Training BOCANet*

To train our BOCANet scheme, we consider a small number of input/output observations taken from an activated IC from a huge pool of all possible I/O patterns. For instance, if a benchmark has N inputs, 2^N different input stimuli are possible. We assume that the attacker has access to a limited and insignificant number of I/Os (less than 0.5% of the data). Deep recurrent neural network is trained by *load profile batches* randomly fetched from the load profile pool so that D-RNN is not only learning individual load patterns but also the common sharing load features and uncertainties. All layers of the networks, except for the last ones which contain linear neurons, are made of sigmoid type neurons and have a neuron for bias. We adjust the weights of the neural network to minimize the MSE on training set. In order to prevent local minimum and overfitting, recursive weight coefficients within a layer of back-propagation are chosen as appropriate batch size. In fact, in each of D-RNN's training iterations, the training batch is firstly fetched from the data pool, and then fed into the D-RNN network. Each training batch includes two matrices with fixed size, i.e., input matrix with size $B \times I$ and output matrix with size $B \times O$. The time-cost and iteration of training process highly depend on the feed-in data sequence size I, the choice of optimizer, the network size (I, H), and the training batch size B. We do not employ a pre-training step; deep architectures are trained from scratch with the supervised mean square error of network. Additionally, we operate early stopping: Out of all iterations, the model with the best development set performance is picked as the final model to be evaluated.

*1) **Attack1-Key Reconstruction**:* In this attack model, the attacker's objective is to reconstruct the secret key from logic encryption obfuscation circuit. As indicated earlier, using D-RNN, two phases are needed for this attack model. In the training phase, we train input and output pairs without providing any key to neural network. Once D-RNN's parameters and weights are trained, we apply key as an input value. Here, initially, the key value is assigned randomly through BOCANet. Next, during the test phase, the initial key value will be later updated based on the MSE of the trained outputs and the newly-generated outputs due to the presence of secret key in the neural network. We have evaluated this attack on 4 different ISCAS-85 benchmarks with different size of keys (32-bit, 64-bit, 128-bit, and 256-bit). The information regarding the benchmarks are shown in Table I. As seen in this table, we were able to reconstruct the secret key (32 bits) from benchmark ISCAS-85 c423 in 6 minutes from BOKANet *that is at least 20 times faster than SAT attack*. For the second benchmark (ISCAS-85 c1355) which has the key size of 64 bits, prediction duration for the key took only 11 minutes. ISCAS-85 c1908 was the third benchmark that we have tested for this attack. This benchmark has the key size of 128 bits and our BOCANet scheme needed only 19 minutes to break this 128-bit secret key. The last benchmark (ISCAS-85 c7552) has key size of 256 which is very large and, again, our proposed attack using D-RNN could break the key in almost 35 minutes. This benchmark has much higher number of inputs and outputs and key size compared to the other 3 benchmarks. Still, our result is much better than that of SAT attack which takes a few hours to break reasonably-large key sizes. Fig. 6 shows the tradeoff between the training size and successful rate of guessing secret keys for each benchmark. As can be seen, by increasing the training size to 100K, the successful rate goes up. Note that we applied BOCANet with one and two hidden layers of D-RNN which is shown in the figure as "1HL" and "2HL", respectively. As seen in Fig. 6, the success rate of guessing secret keys (using two hidden layers) for logic encryption obfuscation benchmark1 is 100%, benchmark2 is 94%, benchmark3 is 92%, and for benchmark4 is 89%. As a result, by having two hidden layers, we achieved higher success rate.

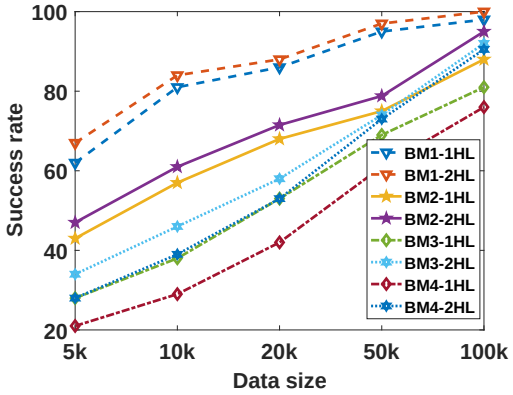
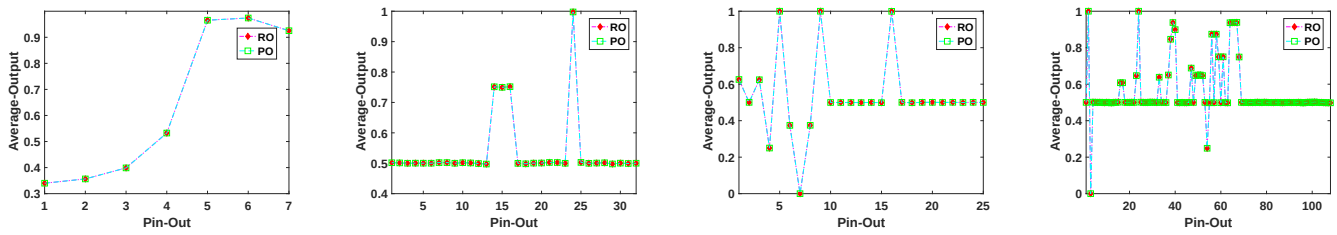


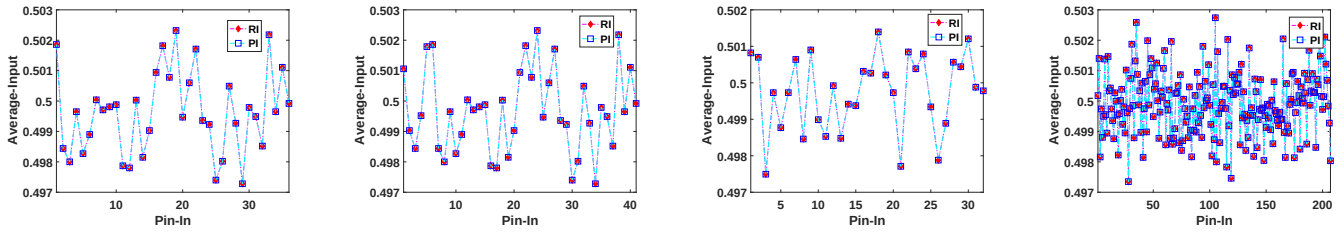
Figure 6. Success rate of BOCANet attack for 4 benchmarks based on one and two hidden layers.

*2) **Attack2-Output Guessing**:* This attack scenario has two phases: i) Training phase: During the training phase, the D-RNN is supervised to learn from the data by presenting the training data at the input layer and dynamically adjusting the parameters of the D-RNN to achieve the desired output value from the input set. Basically, an attacker trains I/O pairs to train D-RNN parameters. Once the parameters are trained, by giving new input patterns, D-RNN will be able to predict



(a) (b) (c) (d)

Figure 7. Average prediction of output patterns from four ISCAS benchmarks (a) c423, (b) c1355, (c) c1908, and (d) c7552.



(a) (b) (c) (d)

Figure 8. Average prediction of input patterns for four ISCAS benchmarks (a) c423, (b) c1355, (c) c1908, and (d) c7552.

the corresponding output stimulus. In this scenario, we train only less than half a percent (¡ 0.5%) of the total available I/O patterns. ii) Testing phase: In order to evaluate our attack model, we test D-RNN with one million new untrained input patterns to predict the outputs. Next, we compare the predicted outputs with the desired ones to ensure the success of our attack model. This scenario is applied to all 4 different ISCAS-85 benchmarks with various number of inputs and outputs. The results of our work is shown in Fig. 7. In this figure, x-axis and y-axis show the number of outputs (pin-out) and average value of the corresponding output pin, respectively. Note that average value here is the average possible outputs from each benchmark. Moreover, RO (red line) indicates "real output" and PO (green line) denotes the "predicted" output from Deep-RNN. Fig. 7(a) is the result for benchmark1 (ISCAS-85 c423) that clearly demonstrates that these two lines (RO and PO) are overlapped; meaning D-RNN could predict the output patterns for this benchmark successfully. Similar scenario is seen for Figs. 7(b-d) which obviously have more output patterns to be predicted from benchmark2 (ISCAS-85 c1355), benchmark3 (ISCAS-85 c1908), and benchmark4 (ISCAS-85 c7552). Note that in this attack scenario, D-RNN does not need a secret key to predict output patterns.

*3) **Attack3-Input Guessing**:* This attack model is very similar to the concepts that we have defined in Attack2 except that the inputs will be predicted by considering recursive D-RNN. In this scenario, an attacker applies desired outputs to predict the input stimuli. The results of our work are shown in Fig. 8. In this figure, x-axis and y-axis show the number of inputs (pin-in) and average value of the corresponding input pin, respectively. Similar to the previous attack, Fig. 8(a) is the result for benchmark1 (ISCAS-85 c423) demonstrating that these two lines (RI and PI) are overlapped; meaning D-RNN could predict the input patterns for this benchmark successfully. Similar scenario is seen for Figs. 8(b-d) (have more input patterns to be predicted from benchmark2 (ISCAS-85 c1355), benchmark3 (ISCAS-85 c1908), and benchmark4 (ISCAS-85 c7552)). Similar to Attack2 scenario, in this attack model, D-RNN does not need a secret key to predict the output patterns.

V. CONCLUSION AND FUTURE WORK

This paper, for the first time, explores the potential of employing the state-of-the-art deep learning technique (Deep RNN) that allows an attacker to derive the correct values of the key inputs (secret key) from logic encryption hardware obfuscation techniques. We have also presented "output-guessing" and "input-guessing" attacks in which an attacker can guess outputs and inputs without having the knowledge of secret key, respectively. Our result indicates that the proposed method can deliver significant improvement for breaking logic encryption. Compared with state-of-the-art attacks, the proposed method (BOCANet) is 20 times faster with relatively-high success rate using only a small number of input/output observations (0.5%) taken from an activated IC. We believe that with augmenting hardware attacks on obfuscated architectures by incorporating deep learning in our scheme not only a practical paradigm shift to attack such obscured circuits is proposed but also it can be utilized as a step forward towards finding the vulnerabilities of other hardware security approaches. As future work, we will investigate (i) if the proposed approach would make delay+logic locking (DLL) based countermeasures more effective and (ii) if the presented scheme could be augmented by such countermeasures to introduce even more powerful mechanisms. We also intend to develop side-channel attack-based mechanisms, e.g., through first and higher order

differential power analysis attacks, to make the proposed schemes, potentially, even more powerful.

REFERENCES

[1] D. Forte, S. Bhunia, and M. M. Tehranipoor, *Hardware protection through obfuscation*. Springer, 2017.

[2] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against ic counterfeiting and overproduction," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 9.

[3] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 137–143.

[4] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 953–958.

[5] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in ic piracy with test-aware logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, 2015.

[6] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating sat-unresolvable circuits," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*. ACM, 2017, pp. 173–178.

[7] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[8] 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 *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 189–210.

[9] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[10] R. S. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2008, pp. 674–677.

[11] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.

[12] J. A. Roy, F. Koushanfar, and L. Igor, "Markov. 2008. epic: Ending piracy of integrated circuits," in *Proceedings of the Conference on Design, Automation, and Test in Europe*, pp. 1069–1074.

[13] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 127–146.

[14] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 236–241.

[15] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Appsat: Approximately deobfuscating integrated circuits," in *Hardware Oriented Security and Trust (HOST), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 95–100.

[16] "Trust-Hub," https:/trust-hub.org/OBFbenchmarks.html, 2017.

[17] C. Spear, "A guide to learning the testbench language features," *System Verilog for verification, 2nd ed., Springer Publishing Company, Incorporated*, pp. 11–18, 2008.

[18] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.