DeepAttack: A Deep Learning Based Oracle-less Attack on Logic Locking

Anand Raj*, Nikhitha Avula*, Pabitra Das*, Dominik Sisejkovic†, Farhad Merchant†‡, and Amit Acharyya*§
*Department of Electrical Engineering, Indian Institute of Technology, Hyderabad, India
ee21resch01012@iith.ac.in, pabitra.das@ee.iith.ac.in, amit_acharyya@ee.iith.ac.in
† RWTH Aachen University, Germany, dominik.sisejkovic@rwth-aachen.de
‡ Newcastle University, UK

2023 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-6654-5109-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISCAS46773.2023.10182222

***Abstract*—Logic locking is one of the most promising design-for-trust technique for protecting intellectual property from reverse engineering, IP piracy, and modification throughout the electronic supply chain. However, oracle-less deobfuscation attacks that do not require an activated chip have been successful in obtaining the secret key of locked designs. This requires a detailed determination of the extent of vulnerability available in obfuscated circuitry. In this paper, we propose the oracle-less *DeepAttack*: an attack on logic locking that is capable of extracting the activation key of the locked netlist using a deep learning model. Based on the ISCAS-85 and EPFL benchmarks evaluation, DeepAttack achieves an average key prediction accuracy of 93.39%, outperforming the oracle-less state-of-the-art attacks SAIL, SnapShot, and OMLA by 21.28, 10.73, and 3.84 percentage points, respectively.**

***Index Terms*—Logic Locking, Hardware Security, Reverse Engineering, IP Protection, Deep Learning.**

I. Introduction

Over the last two decades, the manufacturing cost of Integrated Circuits (ICs) has increased with increasing chip design complexity. The expenditure of establishing a semiconductor fabrication facility lies between $15-$20 billion USD [1]. Hence, it has become very difficult for a small organization to design, test, and fabricate ICs under one (trusted) roof. Thus, many Intellectual Property (IP) owners are forced to outsource the IP design and fabrication to third parties to stay competitive and lower the production cost. Unfortunately, this business model exposes the IP to untrusted parties throughout the IC supply chain. Thus, a multitude of security threats are potentially enabled throughout the IC supply chain, including reverse engineering, IP piracy, hardware Trojan insertion, counterfeiting, overproduction, and key retrieval [2], [6]. It has been estimated that billions of USD are lost every year due to IC theft and design piracy [7] issues.

Different Design-for-Trust (DfTr) techniques were developed to ensure trust in the supply chain, including IC metering [8], [9], watermarking [10], camouflaging [11], [13], split manufacturing [15], [17], and logic locking [18], [21]. Among the DfTr techniques, logic locking is considered the most promising candidate for the protection of IP throughout the IC supply chain [14], [19]. However, different methodologies have successfully attacked logic locking thereby exploiting a set of functional and structural features of locking policies. All attacks on logic locking can be divided into two categories: oracle-guided and oracle-less. Oracle-guided attacks can extract the correct key of an obfuscated design by guiding the attack process with golden I/O patterns obtained from an activated (oracle) IC. Even though effective, oracle-guided attacks are often placed in unrealistic scenarios because of the availability of physical attacks on the oracle, which diminish the need for an oracle-guided process [16]. Thus, at the moment, oracle-less attacks, which do not need the golden reference, fulfill the needs of a more realistic attack model. Recently, many oracle-less attacks, including SAIL [23], SnapShot [24], and OMLA [25], have been proposed [4]. These attacks utilize machine learning to extract the correct key from the locked netlist without using a golden reference. Even though the attacks demonstrated excellent accuracy, they still impose some limitations. SAIL requires a pre-synthesis netlist for training, which may not be readily available to attackers, and it depends on multiple learning models. Moreover, SAIL attack is limited to XOR/XNOR-based locking. SnapShot and OMLA generate individual models to attack a particular benchmark circuit. Hence, for the key prediction of seven benchmarks, seven individual models are generated. Although the SnapShot model can find a suitable neural network automatically, it depends on the performance of the genetic algorithm. To overcome the challenges described above, we propose a deep learning-based, oracle-less attacking methodology. The main contributions of this work are as follows:

- DeepAttack: a deep neural network-based single generic attack model. A detailed description of the attack framework is given, beginning from logic locking and subgraph generation to key prediction.
- An evaluation of DeepAttack on ISCAS-85 and EPFL benchmark circuitry to validate its efficiency and generic nature. The results indicate that DeepAttack is capable of reaching an average Key Prediction Accuracy (KPA) of 93.39%, outperforming the state-of-the-art SAIL, SnapShot, and OMLA attacks by 21.28, 10.73, and 3.84 percentage points, respectively.

§Corresponding Author

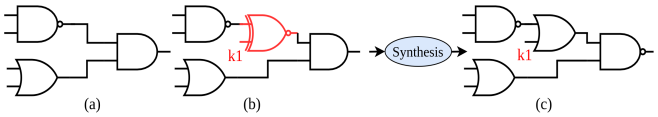

Fig. 1. Example of logic locking: (a) original design, (b) locked with XOR/XNOR gate, and (c) re-synthesized design.

II. DeepAttack

In this section, we discuss the step-wise process flow of the proposed DeepAttack to determine the original key from the obfuscated netlist. However, before we introduce the proposed model, it is important to briefly discuss the state-of-the-art logic locking algorithms and locality extraction.

A. Logic Locking

The primary function of hardware obfuscation or logic locking is to hide the design intent and prevent black box uses from reverse engineering by the untrusted entity. Obfuscation is done by inserting a logic cell in the design. An obfuscated logic cell can be any specific design or standard cell that is activated upon entering a particular key input pattern. Usually, two types of logic locking are used for protecting design: (a) combinational and (b) sequential. For combinational logic locking, key-dependent combinational standard cells, such as XOR/XNOR, AND, MUX gates, are inserted into the design. The functionality of the design is correctly obtained only when the correct key is given to the obfuscated cell. Sequential logic locking is based on the obfuscation of the state space of the design. In this paper, we only focus on XOR/XNOR-based logic locking [26], however, the attack can also be applied to MUX-based locking policies.

Fig. 1(a) shows the original design and Fig. 1(b) the XNOR-gate locked design. The design will function correctly only if *k1=1*. Re-synthesis is performed on the locked netlist to camouflage the inserted key gate, as depicted in Fig. 1(c). Thus, the attacker is not able to simply guess the originally inserted key gate. Based on the statistical analysis of the obfuscated netlist after re-synthesis, *it can be observed that structural changes on the obfuscated netlist near the root node are very sparse, deterministic, and local* [20], [23]. Hence, machine learning can learn to predict the correct key based on the available locality information. Hereby, the locality describes a subgraph around a selected key gate.

B. Attack Methodology

Fig. 2(a) presents the overview of the proposed attack for key prediction using deep learning networks. The ISCAS-85 benchmarks are first compiled into a gate-level description. Afterward, logic locking is deployed to the netlists, resulting in a locked gate-level netlist. Re-synthesis is performed on the locked netlists to acquire the post-re-synthesis locked netlists. The re-synthesized locked netlist is then converted to a graph. A subgraph is extracted for each key bit from the graph in the form of a vector, resulting in a locality, as shown in Fig. 3. The extracted localities are separated into two classes, i.e.,

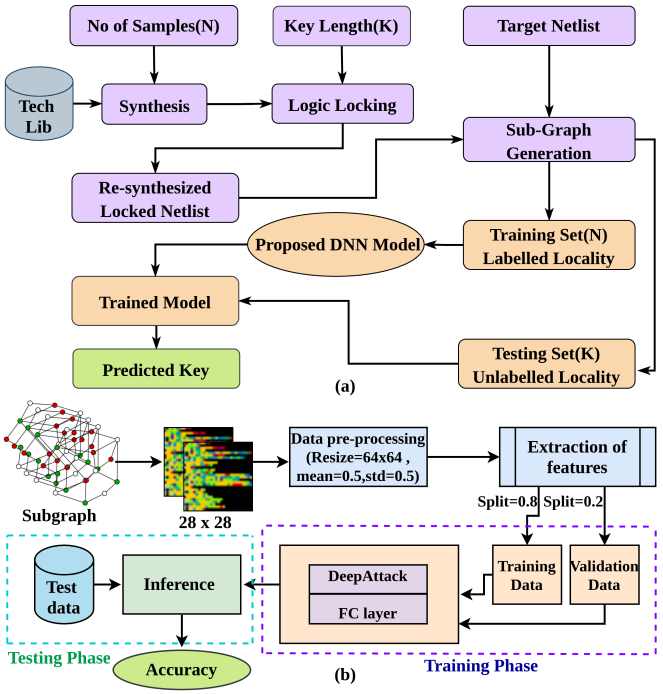

Fig. 2. Overview of the DeepAttack methodology: (a) top-level attack flow and (b) workflow of the DNN architecture.

zero-labeled and one-labeled localities based on the type of key value (0 or 1). Zero-labeled and one-labeled vectors are stacked horizontally to form two images. After analyzing zero-labeled and one-labeled images, we can observe that there is a repeating pattern corresponding to each key bit value, as confirmed in [24]. This visual difference proves that even humans can perform the classification. Since extracting and learning correlation in data is a key feature of Convolutional Neural Networks (CNNs), it stands to reason that CNNs are a suitable choice for this classification problem. The extracted localities are transformed into labeled 28x28x1 images and fed to the proposed deep neural network model for training. ResNet [27] is one of the most efficient Deep Neural Network (DNN) architectures for classification, as it helps maintain a low error rate much deeper in the network, thus resulting in a higher accuracy compared to simple CNNs [24].

For testing, the target netlist's localities are extracted in the same way as the training data, only this time without the corresponding (key) label. The unlabeled images are fed to the proposed DNN model for key prediction. The key prediction accuracy describes the percentage of correct keys predicted among all keys. Thus, the main objective of the attack is to reach a high KPA.

The proposed DNN model has 18 layers composed of 8 residual blocks. The DNN workflow is shown in Fig. 2(b). In the data pre-processing stage, we resize the images to (64 $\times$ 64) and normalize the image by setting the mean and standard deviation to 0.5. Data is split into train and validation in the ratio of 8:2. Split data is fed to the proposed model, having

TABLE I
HYPERPARAMETERS FOR THE DNN MODEL.

Parameter	Value
Input/Conv Act. Func.	ReLU
Output Layer Act. Func.	Softmax
Loss Functions	Crossentropy
Optimizer/Learning Rate	RMSprop/0.001
Batch Size	64
Epochs	50

18 layers, 1st layer of (7×7) kernel size, 16 convolution layers with (3×3) kernel size and 8 residual blocks. The residual block uses skip connections, which enables the DNN model to learn features efficiently. The output is obtained from a fully connected layer with (512, 2) in and out features. Cross entropy and RMSprop are used as loss functions and optimizers to train the model, where the learning rate is set to 0.001 for 50 epochs. Total trainable parameters are 11,177,538 accounting for a model size of 42.64MB. Model performance is evaluated using a test set consisting of images formed from the target netlists. All the hyperparameters used for tuning the model are listed in Table I.

DeepAttack is deployed to predict the key for the target netlist. The features (Z_i) are obtained after passing the test set to the DNN model. The function $H = \frac{exp(Z_i)}{\Sigma_j Z_i}$ returns the probability of each class (either *key 0* or *key 1*). Passing features to the above equation, we obtain prediction probabilities of each class, and further converting to percentage is explained in Algorithm 1. Key is predicted based on comparing *key 0* and *key 1* prediction percentages. If the prediction percentage of *key 0* is greater than *key 1* then the predicted key is 0 and vice versa.

Algorithm 1 Key Prediction

Input: Z_{ix} : Test Image Features
Output: Key
  for x=0,1 **do**
    $P_{ix} \leftarrow H(Z_{ix})$ {Returns probability}
    $Predict[P_{ix}] \leftarrow 100 \times P_{ix}$
  end for
  if $P_{i0} > P_{i1}$ **then**
    $key \leftarrow 0$
  else if $P_{i0} < P_{i1}$ **then**
    $Key \leftarrow 1$
  end if
  return Key

Locality extraction is the process through which local information from the re-synthesized locked netlist for each key gate or root node is extracted. Note that there is a multitude of ways to extract a locality. In this work, we perform the extraction based on the process proposed in [24]. The re-synthesized netlist is converted into a directed graph. All gates are encoded with a number that uniquely identifies the gate type. From the graph, we extracted a subgraph for each root node using the Breadth-First Search (BFS) algorithm, as shown in Fig. 3.

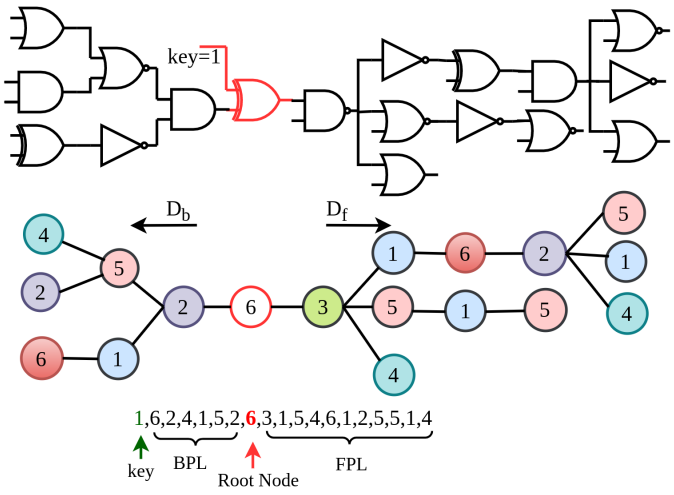

Fig. 3. Example of locality extraction from the locked re-synthesized netlist.

After the analysis of the structure, we concluded that a depth level of 5 towards the primary outputs ($D_f = 5$) and a depth level of 3 towards the primary inputs ($D_b = 3$) is required to capture all synthesis-induced local changes around a key gate. From the subgraph, we generate local information in the form of a locality vector. Hence, for every key, there will be one vector. The vector has information on the root node, Forward Path Locality (FPL), Backward Path Locality (BPL), and key information. Note that the key value is only known for the (labeled) training data.

III. RESULTS AND ANALYSIS

In this section, we present and discuss the evaluation of DeepAttack on ISCAS-85 benchmark circuits. Circuits C1355, C1908, C2670, C3540, C5315, C6288, and C7552 were locked 1200 times with a 64-bit key to get 537600 training localities. The data is randomly split using an 80:20 ratio for training and validation. During testing, K localities were extracted around the key gates from the originally locked circuits (unknown key), and an unlabeled test set is formed and fed to the DNN architecture to predict the correct key. DeepAttack is compared to state-of-the-art methodologies SAIL, SnapShot, and OMLA. DeepAttack has a single DNN model which can predict the key for all benchmark circuits. One of the advantages of DeepAttack over all previous attacks is that all attacks rely on a different model for different benchmark circuits. Therefore, to attack seven benchmarks, seven models are required [24], [25].

The KPA results of DeepAttack compared to SAIL, SnapShot and OMLA on ISCAS-85 benchmarks are presented in Fig. 4. DeepAttack achieves an average KPA of 93.39% outperforming SAIL, SnapShot, and OMLA attacks. DeepAttack is able to predict the key of C1355, C1908, C2670, C3540, C5315, C6288, and C7552 with 90.9%, 86.36%, 95.65%, 90.9%, 100%, 98.65%, 91.3%, respectively. C5315 has reached accuracy up to 100%. The high accuracy across

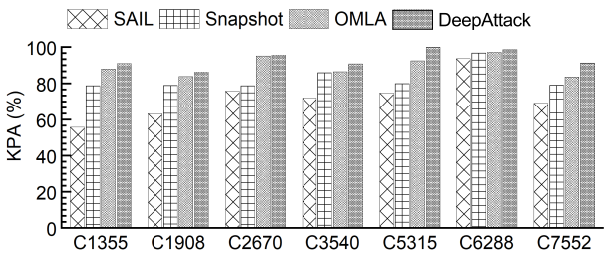

Fig. 4. Evaluation results on ISCAS-85 benchmark circuits.

TABLE II
IMPACT OF THE NUMBER OF LOCKED SAMPLES ON ACCURACY.

Circuits	500 Locked	1000 Locked	1200 Locked	1500 Locked	2000 Locked
C1355	80.95	85.71	90.9	90.90	90.92
C1908	54.5	72.72	86.36	72.72	72.74
C2670	95.45	95.45	95.65	100	100
C3540	68.20	81.81	90.90	90.90	90.90
C5315	86.90	95.83	100	95.65	95.65
C6288	68.18	95.65	98.65	98.65	98.67
C7552	68.2	86.36	91.3	100	100
Avg. KPA	**74.62**	**87.65**	**93.39**	**92.68**	**92.69**

different benchmarks indicates that DeepAttack's accuracy does not depend on the circuit type or size.

To check the implications of the number of locked samples that are used for training and testing, we locked all seven benchmarks 500, 1200, 1500, and 2000 times with 64-bit keys, resulting in 22400, 448000, 537600, 672000, and 896000 localities, respectively. Table II shows the impact of the number of locked samples (repetitions) on accuracy. We can observe that for 500 locking rounds, the average accuracy is 74.20%, which is less than all other configurations. This shows that a lower number of samples generates less variation in locality information. For 1200 repetitions, the average KPA is 93.39%, which is greater by 21.28, 10.73, and 3.84 percent points compared to SAIL, SnapShot, and OMLA. Furthermore, we observe that the average accuracy for 1500 and 2000 repetitions is 92.68% and 92.69%, respectively. This indicates that the training set gets saturated with already existing localities, thus not having a positive impact on the overall attack's performance. Hence, for our evaluation, 1200 repetitions seem to be the optimal number.

Note that the proposed DeepAttack can work with any type of locking technique, because it is based on the locality information of the locked design near the root node, which can always be extracted in a certain format. Thus, the attack is not limited to random logic locking.

To check the performance of our model in the generic case in which the training and validation set are not based on the same netlists, we have taken training/validation data from the ISCAS-85 benchmark circuit, and we attacked four circuits (Arbiter, Multiplier, Voter, and Memory) from the EPFL benchmark suite. It has been observed that the average key prediction accuracy is 66.58%. The performance gap compared to the self-referencing model is likely caused by the DNN having to make predictions of new localities based on knowledge collected from other benchmarks, rather than the target ones. This effect has also been documented in [24].

The Synopsys design compiler was used for synthesis alongside the Cadence 45nm Generic Library. The evaluation was performed on an Intel Xeon® W-1350 Processor with 32GB of RAM. Graph extraction is done using *Networkx* and *NumPy* library in Python. The proposed deep learning model is trained with the *PyTorch* framework. The average computational runtime of OMLA on the ISCAS-85 benchmark is 1.85h and our proposed model has an average run time of 1.56h. The inference time of OMLA on its own does not take more than 1s for all evaluated benchmarks. (Inference time of each benchmark is 0.16-0.17 ms and accounts 1.12-1.19 ms for all benchmark circuits).

IV. CONCLUSION

In this paper, we proposed a deep learning oracle-less attack called *DeepAttack* that can predict keys without a golden reference. DeepAttack is based on a locality representation that captures netlist information in a vector that describes certain subgraphs around selected key-controlled gates. The evaluation results show that DeepAttack is able to outperform the state-of-the-art attacks SAIL, SnapShot, and OMLA, thereby reaching a key prediction accuracy of 93.39% on average.

V. ACKNOWLEDGEMENT

This work has been supported by the Ministry of Electronics and Information Technology (MEITY), Government of India.

REFERENCES

[1] The Rising Cost of Semiconductor R&D. [Online] "https://www.aalbun.com/blog/the-rising-cost-of-semiconductor-rd
[2] Chipworks. Intel's 22-nm Tri-gate Transistors Exposed. http://www:chipworks:com/blog/technologyblog/2012/04/23/intels-22-nm-tri-gate-transistors-exposed/, 2012.
[3] M. Rostami, F. Koushanfar, and R. Karri, A Primer on Hardware Security: Models, Methods, and Metrics, Proceedings of the IEEE, 102, 1283–1295, 2014.
[4] D. Sisejkovic, L. M. Reimann, E. Moussavi, F. Merchant and R. Leupers, "Logic Locking at the Frontiers of Machine Learning: A Survey on Developments and Opportunities," 2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC), Singapore, Singapore, 2021, pp. 1-6.
[5] R. Torrance and D. James, The State-of-the-Art in Semiconductor Reverse Engineering, IEEE/ACM Design Automation Conference, 333–338, 2011.
[6] M. M. Tehranipoor, U. Guin, and S. Bhunia, Invasion of the Hardware Snatchers, IEEE Spectrum, 54, 36–41, 2017.
[7] Estimating the global economic and social impacts of counterfeiting and piracy 2011 [online] https://iccwbo.org/publication/estimating-global-economic-social-impacts-counterfeiting-piracy-2011/
[8] Y. Alkabani and F. Koushanfar, Active Hardware Metering for Intellectual Property Protection and Security, USENIX Security, 291–306, 2007.
[9] F. Koushanfar, Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management, IEEE Trans. Inf. Forensics Security, 7, 51–63, 2012.
[10] A. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. Markov, et.al., Watermarking Techniques for Intellectual Property Protection, IEEE/ACM Design Automation, 776–781, 1998.
[11] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, Security Analysis of Integrated Circuit Camouflaging, ACM/SIGSAC Conference on Computer & Communications Security, 709–720, 2013.

[12] M. Massad, S. Garg, and M. Tripunitara, Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes, Network and Distributed System Security Symposium, 2015.
[13] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, CamoPerturb: Secure IC Camouflaging for Minterm Protection, IEEE/ACM International Conference on Computer-Aided Design, 29:1–29:8, 2016.
[14] Sisejkovic, D., Merchant, F., Reimann, L.M., Leupers, R., Kegreiß, S. (2020). Scaling Logic Locking Schemes to Multi-module Hardware Designs. In Architecture of Computing Systems – ARCS 2020, vol 12155., Springer, Cham.
[15] R. Jarvis and M. McIntyre, Split Manufacturing Method for Advanced Semiconductor Circuits, US Patent 7, 2007.
[16] Sisejkovic, D., and Leupers, R. (2023). Working Principle and Attack Scenarios. In: Logic Locking. Springer, Cham.
[17] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation, USENIX Conference on Security, 495–510, 2013.
[18] J. Roy, F. Koushanfar, and I. L. Markov, Ending Piracy of Integrated Circuits, IEEE Computer, 43(10),30–38, 2010.
[19] Tan, B., Karri, R., Limaye, N., Sengupta, A., Sinanoglu, O., Benchmarking at the frontier of hardware security: Lessons from logic locking. arXiv preprint arXiv:2006.06806.
[20] D. Sisejkovic, F. Merchant, L. M. Reimann and R. Leupers, "Deceptive Logic Locking for Hardware Integrity Protection Against Machine Learning Attacks," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 6, pp. 1716-1729, June 2022.
[21] Sisejkovic, D. and Leupers, R. (2023), Logic Locking: A Practical Approach to Secure Hardware, Springer, Cham..
[22] Subramanyan, P., Ray, S. and Malik, S., May. Evaluating the security of logic encryption algorithms. IEEE International Symposium on Hardware Oriented Security and Trust (HOST), 137-143, 2015.
[23] Chakraborty, P., Cruz, J. and Bhunia, S., December. SAIL: Machine learning guided structural analysis attack on hardware obfuscation. Asian Hardware Oriented Security and Trust Symposium (AsianHOST), 56-61, 2018.
[24] Sisejkovic, D., Merchant, F., Reimann, L.M., Srivastava, H., et.al., Challenging the security of logic locking schemes in the era of deep learning: A neuroevolutionary approach. Emerging Technologies in Computing Systems (JETC), 17(3), 1-26, 2021.
[25] Alrahis, L., Patnaik, S., Shafique, M. and Sinanoglu, O., OMLA: An oracle-less machine learning-based attack on logic locking. IEEE Transactions on Circuits and Systems II, 69(3), 1602-1606, 2021.
[26] Yasin, M., Rajendran, J.J., Sinanoglu, O. and Karri, R., On improving the security of logic locking. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 35(9), 1411-1424, 2015.
[27] He, K., Zhang, X., Ren, S. and Sun, J., Deep residual learning for image recognition. IEEE conference on computer vision and pattern recognition, 770-778, 2016.