文章编号:2095-6134(2021)04-0494-09

Efficient machine learning methods for hardware Trojan detection using instruction-level power character*

LI Ying[1], CHEN Lan[1,2†], TONG Xin[1]

(1 *Institute of Microelectronics, Chinese Academy of Sciences, Beijing* 100029, *China*; 2 *Beijing Key Laboratory of Three-dimensional and Nanometer Integrated Circuit Design Automation Technology, Beijing* 100029, *China*)

(Received 26 September 2019; Revised 22 November 2019)

Li Y, Chen L, Tong X. Efficient machine learning methods for hardware Trojan detection using instruction-level power character[J]. Journal of University of Chinese Academy of Sciences, 2021,38(4):494-502.

Abstract Integrated circuits (IC) are vulnerable to hardware Trojans (HTs) due to the globalization of semiconductor design and outsourcing fabrication. Stealthy HTs which activate malicious aging operations are ususlly hide in normal behaviors. Therefore, it is a challenge to detect those HTs by general test and verification approaches. In this paper, we build an efficient machine learning (ML) framework to classify the genuine and Trojan-insert chips using instruction-level side-channel power characters. Different instructions and HTs are used as feature sets to construct the algorithm models. In order to evaluate the performance of the method, we implemented five HTs benchmarks of MC8051 micro-controller in Altera Stratix II FPGA, and presented analysis on five formulated ML models in both supervised and unsupervised modes. The test results showed that the detection accuracy of supervised Naïve Bayes is 95% in average, which is the highest among the ML models. The supervised SVM consumed the shortest running time, with an average of 0.04 s. We also verified that one-class SVM can be a valuable method without golden reference, which has accuracy in the range from 17% to 72% even in Harsh learning condition.

Keywords hardware Trojans; machine learning; side-channel power; instruction-level; detection

CLC number:TN406 **Document code**: A **doi**:10.7523/j.issn.2095-6134.2021.04.008

指令级功耗特征的硬件木马检测高效机器学习

李莹[1],陈岚[1,2],佟鑫[1]

(1 中国科学院微电子研究所, 北京 100029;2 三维及纳米集成电路设计自动化技术北京市重点实验室, 北京 100029)

摘 要 由于半导体产业的设计和外包代工制造全球化趋势,使得集成电路容易受到硬件木马造成的严峻威胁。基于电路退化模型等的隐秘硬件木马通常将恶意行为隐藏在正常的芯片

* Supported by Beijing Natural Science Foundation (4184106), National Internet of Things and Smart City Key Project Docking (Z181100003518002), Beijing Science and Technology Project (Z171100001117147)

† Corresponding author, E-mail: chenlan@ime.ac.cn

行为中,从而难以被传统的测试和验证方法发现。建立一个高效的机器学习框架,利用指令级侧信道功耗特征对无木马和插入木马的芯片电路进行分类。算法模型采用不同的指令和木马构造提取的特征向量集。为评估检测方法性能,在 Altera Stratix Ⅱ FPGA 中实现基于 MC8051 微控制器的基准电路,并详细分析在有监督和无监督模式下的 5 种机器学习算法模型。测试结果表明,综合各种特征条件,有监督的朴素贝叶斯方法检测准确率最高,平均为 95%,有监督的支持向量机方法运行时间最短,平均为 0.04 s。另外验证了无监督的支持向量机可以作为一种没有黄金参考模型下的有价值方法,即使在恶劣训练条件下,其检测准确率也在 17% ~72%。

关键词　硬件木马;机器学习;旁路功耗;指令级;检测

The trend of globalization, outsourcing and split fabrication give the attackers more opportunities to tamper the IC design with hardware Trojans (HTs). Such malicious circuits can be implanted either during the design or manufacturing phase, which enable the adversary to spy confidential contents, control, monitor kernel functions or deny service in systems[1].

Since in 2005, DARPA issued its 1st program for hardware systems security, many HTs detection techniques have been proposed. Nondestructive methods, especially side-channel parameter measurements have received a lot of attention[2-13]. However, in a system chip, the activating impact of HTs under certain pattern can be so small that hide in normal functions. The stealthy ones with aging triggers, which control the lifetime of a circuit by counters or timers and violate runtime operations, can even bypass the normal verification phase[8-9]. All of the above reasons significantly overwhelmed the performance of side-channel detection. Thus, efficient detection methods from system operational level should be considered.

This paper introduces a solution by using machine learning (ML) algorithms to learn side-channel power characters in instruction-level and classify the genuine and Trojan-insert circuits. The paper mainly contributes as:

1) A creative framework includes feature set generation to extract learnable instruction-level power character, and circuits classification flow using ML models.

2) Comprehensive detecting performance evaluation for both supervised and unsupervised ML modes in terms of the effects of features and time consuming.

3) Fully implementation and case study of a MC8051 micro-controller on Altera FPGA with open source HTs benchmarks.

1　Related works

Nondestructive HTs detection approaches performed at design and test time and can be classified into: 1) Logic testing, which depends on rare conditions and tests the effect of HTs in logic values on outputs[10-11]. 2) Side-channel analysis, which is based on side-channel parameters including transient signals[2], leakage currents[3-4], timing delay[12], regional supply currents[13], electromagnetic radiation[5], as well as multi-parameter combinations[14] to identify malicious modifications in design. However, triggering complex or mix-signal Trojans increase the challenge to use methods in logic testing. For other approaches using side-channel data, the effectiveness is questioned when dealing with stealthy aging Trojans which target to violate runtime operations[15].

Therefore, researchers proposed works to do lifetime HTs testing by adding build-in sensor in circuit[16], or managing dynamic thermal distribution[17]. These techniques monitored specific properties under specific conditions, which required precise calibration to match the environmental changes. In recent past, machine learning algorithms attracted wide attentions from industry and academia in the context of efficient HTs detection. Jap et al.[18] used support vector machine (SVM) and unsupervised model to detect leakage of

AES by EM measurement. Bao et al.[19] classified the IC images of benchmark circuits with K-Means clustering and SVM. Lodhi et al.[20] trained the timing signature with four different algorithms. Xue et al.[21] provided a classification-based detection technique with error weight-adjusting and cost balance. Tomotaka et al.[22] compared the classification results of SVM on Hardware Trojan with and without trigger circuits. All of these works either targeted to stand alone IPs or separated benchmark circuits. Cases involving firmware processing can be extremely different, which still have a lot of open topics in HTs detection realm, especially in features extraction and classifying method selection. Lodhi et al.[23] proposed a method using instruction-level power profile to classify chip behaviors at run-time test but did not consider the effect of various feature conditions in their method and lacked comparison of different ML algorithms.

2 Detection methodology and feature set generation

2.1 Side-channel HTs detection

In this paper, we apply the IDDT-IDDQ method to reduce the impact of both intra-die and inter-die process variations (PV) in detection[2-3,14]. Due to the principle of methodology, we assume the presence of a golden power model of a genuine chip (or layout) in all ML models except one-class SVM. The paper focus mainly on the instruction-level behavior and test efficiency.

Another assumption is that the HTs adds/removes digital logics without violating the chip specification. This is rational because all the Trojan benchmarks we used are well designed and inserted in internal RTL netlists. Therefore, the power introduced by HTs is independent from noise and genuine current[3,14]. Since the noise can be minimized by using Monte Carlo method, the differences in target chip exist only in different test features.

2.2 Feature set generation

In order to overcome the aforementioned shortcomings and to exploit the feature dependencies in ML algorithms, feature sets are generated in order to extract power character.

1) Instruction Difference: The first feature is instruction-type, which determines the basic operation. The most typical 21 instructions (in 7 types) of MC8051 micro-controller with different operands are selected[24], as shown in Fig. 1.

Type1 NOP: No operation					
1	NOP				
Type2 MOV: Move memory, copy operand2 into operand1					
2	MOV_A_RR	3	MOV_A_D	4	MOV_A_DATA
5	MOV_RR_A	6	MOV_RR_D	7	MOV_RR_DATA
8	MOV_D_A	9	MOV_D_RR	10	MOV_D_DATA
Type3 ADD: Add accumulator, add the value of operand to accumulator and store result in accumulator					
11	ADD_A_RR	12	ADD_A_D	13	ADD_A_DATA
Type4 SUBB: subtract from accumulator with borrow					
14	SUBB_A_RR	15	SUBB_A_D	16	SUBB_A_DATA
Type5 INC: Increment operation by operand					
17	INC_A	18	INC_D	19	INC_RR
Type6 JMP: Jump to Data pointer + accumulator to the address represented by DPTR					
20	JMP_A_DPTR				
Type7 JNC: Jump to the relative address if the carry bit is not set					
21	JNC				

Fig. 1 Instruction set in use

2) HTs Difference: Since the structure of HTs and the way they attack the circuit can act extremely various behaviors in operations, a learning model needs to measure the differences to enhance its classification. Therefore, the 2nd feature is the HTs type. The Five Trojans benchmarks come from Trust-Hub[25], all of which are low probability runtime activating HTs. The first 3 HTs (HT1-HT3) add extra logics, and the last 2 HTs (HT4) disable/replace some logics of the original design. The detailed descriptions are shown in Table 1.

3 ML models and framework

3.1 Machine learning models initialization

Five typical classification ML models are formulated to learn the power character, including four supervised methods: k-Nearest Neighbors (k-NN), Naïve Bayes (Bayes), AdaBoost with Decision

Table 1　HTs benchmarks

name	description	trigger type	payload type
HT1	MC8051-T200, the Trojan activates the internal timers of 8051 in the idle mode	internal sequential logic	denial of service
HT2	MC8051-T300, the Trojan is triggered when 8051 sends a specific string of data through UART. In order to block receiving any message through UART	internal sequential logic	denial of service
HT3	MC8051-T500, the Trojan trigger detects a specific command, and the Trojan payload replaces specific data after Trojan activation	internal state machine condition	change function
HT4	MC8051-T600, the Trojan disables any jump in algorithms running by the micro-controller	external combination condition	disable function
HT5	MC8051-T700, the Trojan replaces some input data with some predefined data	internal state machine condition	replace function

Tree (AdaBoost-DT), two classes-SVM (SVM-2C) and one unsupervised method: one class-SVM (SVM-1C). They stand for the four main ML theories in outlier detection field: distance based, statistical based, tree based, and SVM. The initialization of the models are:

1) In k-NN classification, the number of nearest neighbors k, distance metric, and classification rule are basic issues in consideration. Euclidean distance is applied as distance metric to measure the differences between data samples. And the majority voting is selected as classification rule. The value of k is determined by the feedback of cross validation.

2) Naïve Bayes classifier is a highly practical Bayesian learning method with assumption of conditional independence. The trained classifier outputs the best result based on posterior probability.

3) DT learning is a basic classifier which uses a predictive model to form observations about an item (branch) and conclusions about its target value (leaf). In our model, an adaptive fitting (AdaBoost) is further applied into training and decision making phase to reduce bias and variance in advance. The value of DT classifier gets from gradient descent in application.

4) SVM is a popular classification method, which finds the separator maximum margin hyperplane of train data. We apply LIBSVM[26] library to calculate the margin. A linear kernel function is implanted to balance the classification result and time consuming, and the parameter *cost* is set to a small number in order to increase the tolerance of miss-classified boundary data. Both the supervised and unsupervised models are built, in order to compare the performances with Golden model or not. In unsupervised SVM-1C, the training stage uses random selected unknown data set, and the testing stage used the rest part.

3.2　Framework

The proposed framework consists of five major stages (as shown in Fig. 2).

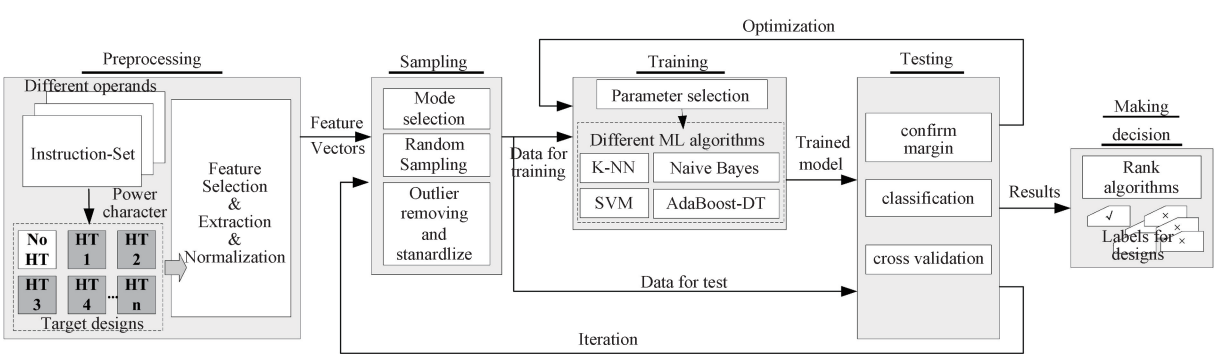

Fig. 2　The proposed framework

3.2.1　Preprocessing

The aim of preprocessing is to obtain feature constrained power character. The main steps include:

1) Apply the instructions as preload commands (test vectors) to the target designs and run implementations separately.

2) Extract and collect respective power data.

Then normalize each character to fit the learning algorithms.

3.2.2 Sampling

The aim of sampling is to treat the extracting characters independently and randomly for valuable model training. The main steps include:

1) Select one mode from the following four: instruction-sensitive (Mode1), HTs-sensitive (Mode2), instruction & HTs-sensitive (Mode3), and none-sensitive (Mode4).

2) In each mode, randomly divide the extracting characters into mutually exclusive n subsets (based on a defined rate). Some sets are treated as training data, others as testing data. For simulation convenience, we mainly use two ways: ① Randomly pick the whole data group from a single HTs benchmark as testing data (Mode2 and Mode3). ② Randomly select a portion of data as testing sample (Mode1 and Mode4).

3) Design an effective Standardization to accelerate convergence, and avoid different feature scales dominating the classification.

3.2.3 Training

Get trained models from the selected vectors using ML algorithms. The main steps include:

a) Set initial value for each parameter, and label the normalized power characters for training as genuine and Trojan-inserted (if necessary).

b) Apply the selected Mode into the ML algorithms and train the learning model separately.

3.2.4 Testing

Evaluate and optimize the trained model using test data with labels. The main steps include:

1) Input test data into the trained model, calculate mathematical results and produce the margin between two classes.

2) Classify the test data based on the above margin in each algorithm.

3) Run m iterations in each sampling case as cross validation. Compare the classification results with known labels. Count the true negative and true positive and calculate the accuracy. If the result exceeds the pre-defined rate, the model will optimize realtive parameters and restart another iteration.

3.2.5 Making decision

This stage is to calculate the overall performance, and provide the final label for each design under test. The main steps include:

1) Rank the algorithms by accuracy and sensitivities to feature conditions.

2) Output the final evaluating decision (Trojan-insert or not) for each design under test.

4 Experimental results and analysis

In order to evaluate the proposed method, we used the EDA-CAS SOC/IP Evaluation Prototype Board v2.0 to do experiments. The FPGA device on the board is Altera Stratix Ⅱ EP2S130F150814 fabricated in 90 nm CMOS technology. The genuine and HTs benchmark circuits of MC8051 micro-controller were implemented on the platform via Quartus Ⅱ version 11.0, separately. The input synchronized clock is 22.42 MHz and the test vectors are the same for all test cases.

The power characters were extracted by using the power analyzing tool PowerPlay in post-gate-level simulation. We did not trigger any HTs in all test cases. Table 2 shows the number of observations in training and test data set in in all test modes.

Table 2 Number of observations in each learning case

		k-NN, Bayes, A-DT		SVM-2C		SVM-1C	
		train	test	train	test	train	test
Mode1	0.1	44	4	29	3	21	11
	0.2	39	9	26	6	19	13
	0.3	34	14	23	9	16	16
	0.4	29	19	20	12	14	18
Mode2		725	145	435	145	290	145
Mode3		40	8	24	8	16	8
Mode4	0.1	783	87	522	58	391	189
	0.2	696	174	464	116	348	232
	0.3	609	261	406	174	304	276
	0.4	522	348	348	232	261	319

We used different sampling ratios in Mode1 and Mode4 (from 0.1 to 0.4) to investigate the performance versus data amount. Moreover, in order to decline the possibility of most training data could come from one same class due to small data amount in Mode1 and Mode3, only those successful running

results account for the accuracy.

The classification results of Mode1 are shown in Fig. 3 (a)-3(e). Each radar figure represents the ML algorithm's detection accuracy of relative instructions, which also can be seen as the different effects of feature 1 (instruction type difference). The accuracy is calculated as number of correctly labeled observations in total labeled observations. All the supervised learning algorithms get satisfied performances, but the unsupervised method only correctly detects different instructions range from 28% to 72%.

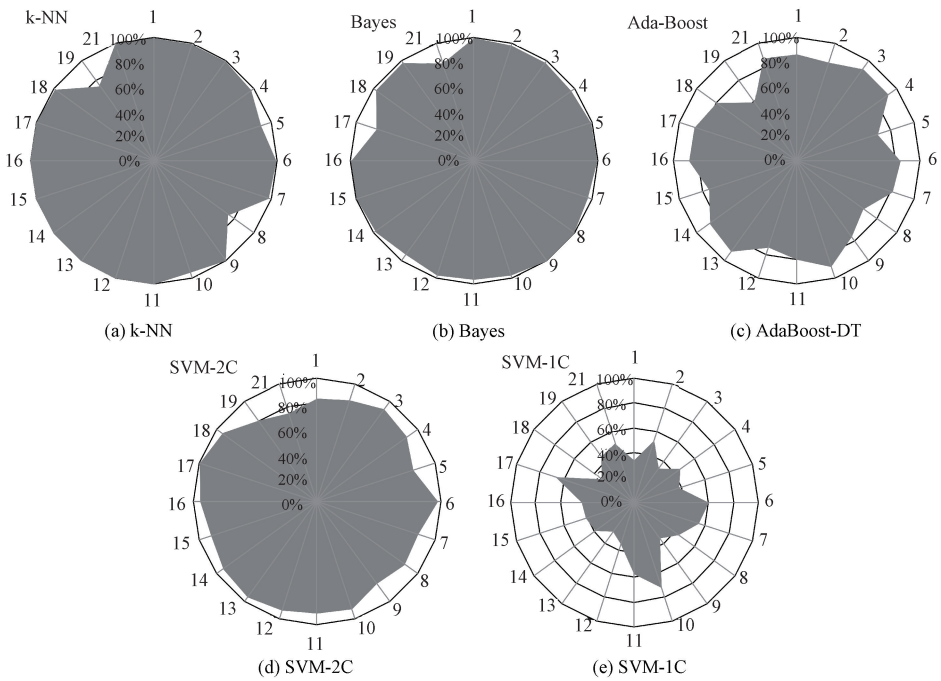

Fig. 3　Detection accuracy of different instructions in Mode1

Figure 4 shows the detection accuracy in relative HTs of Mode2. In the results of HT2-HT5, we obtained 100% accuracy in all supervised algorithms. However, we almost failed in HT1 test except for Bayes method. This is because the offset between HT1 and the genuine chip is very small, which is hard to separate by most classifiers. Since Bayes uses probability instead of distance or boundary to do calculation, it presents the best performance in this mode. The unsupervised SVM-1C can only detect HT3 correctly because it is the largest Trojan circuit.

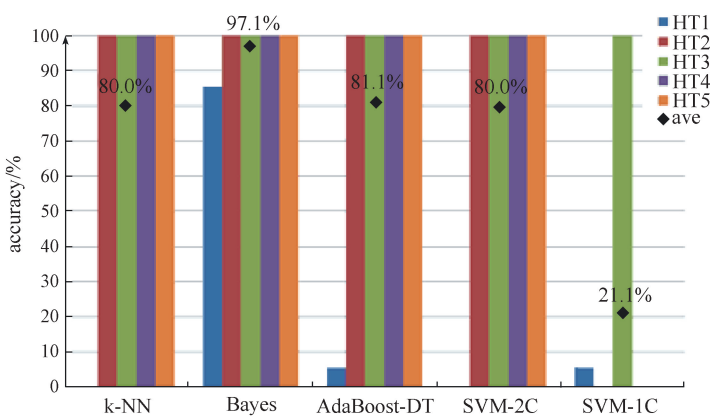

Fig. 4　Detection accuracy of different HTs in Mode2

In instruction & HTs-sensitive mode (Mode3), the combination effect of both instruction and HTs quantifies the mix sensitivity.

We summary the test average accuracy,

instruction sensitivity (Instr-Sens), HTs sensitivity (HTs-Sens), and mix sensitivity of both two features (Mix-Sens) for Mod 1 to Mod 3 in Table 3. From the results, we have the following learnings.

Table 3 Average accuracy and feature sensitivity result for Mode1-3

ML method	Mode1		Mode2		Mode3	
	accuracy/%	Instr-Sens	accuracy/%	HTs-Sens	accuracy/%	Mix-Sens
k-NN	96.6	0.13	80.0	0.45	81.0	0.43
Bayes	96.9	0.05	97.1	0.06	100.0	0
AdaBoost-DT	81.4	0.18	81.1	0.42	80.0	0.44
SVM-2C	89.4	0.11	80.0	0.45	80.0	0.45
SVM-1C	45.5	0.20	21.1	0.44	31.0	0.61

1) Bayes is the least interfered method by both features and gets the highest accuracy in all Modes. Therefore, the Bayes methods can be considered as a premium choice when there is few knowledge about any features.

2) Other supervised methods, including k-NN, AdaBoost-DT and SVM-2C, are more sensitive to HTs types rather than instructions. SVM-1C performs nearly equal to the two features.

None-sensitive (Mode4) can be seen as a rough learning mode, the accuracy in different test sampling ratios is showed in Table 4. In Mode4, since the training and test sets are determined by a random vector, the classification and accuracy results are averaged by five running trails.

Table 4 Detection accuracy in Mode4 %

	test sampling ratio				
	0.1	0.2	0.3	0.4	average
k-NN	96.6	95.9	96.4	96.4	96.3
Bayes	95.2	96.0	95.1	95.1	95.4
AdaBoost-DT	81.6	82.5	82.9	82.9	82.5
SVM-2C	97.9	98.4	97.6	97.6	97.9
SVM-1C	19.5	17.2	53.2	19.8	27.4

In order to compare the performances with reference[23], we infer they performed a coarse sampling process, which is similar with our Mode4. They had accuracy results of 99.02% for k-NN and 86.46% for Bayes, respectively, while both results in our experiment are no less than 95.0%.

According to the result, SVM-2C performs the best in Mode4. Because under one instruction, different power characters produced by different Trojan circuits are more linearly separable, which is the favor condition of SVM.

The computing environment is Intel i5-2400 CPU with 3.10 GHz main frequency. The formalized time consuming in all tests and modes are shown in Table 5. Based on the results, SVM-2C consumes the shortest time to finish the computation, which makes it a competitive option for further hardware implementations.

Table 5 Computing time in all modes s

	Mode1	Mode2	Mode3	Mode4	average
k-NN	0.128 5	0.277 3	0.148 9	0.213 0	0.191 9
Bayes	0.121 6	0.016 8	0.137 3	0.011 0	0.071 7
Ada-Boost	0.584 1	0.102 5	0.620 6	0.084 5	0.347 9
SVM-2C	0.090 6	0.008 9	0.05	0.012 2	0.040 4
SVM-1C	0.130 7	0.070 9	0.115 5	0.090 7	0.102 0

From all of the test results, we can summarize:

1) The power character of a Trojan-insert chip which is close to genuine one rather than other HTs cannot be efficiently detected in k-NN, AdaBoost-DT and SVM (like HT1 in test).

2) The affections of HTs are usually bigger than instructions to all ML methods, and Bayes is proved to be the prospective algorithm in term of accuracy in both instructions and HTs changing situations.

3) Since SVM gets comparable performance and consumes the shortest time to finish the learning loop in average, it can be considered as an efficient hardware model to insert in chips.

4) For one-class SVM, the detection accuracy results in all test modes are much lower than supervised ones due to the influence of uncertain decision boundary from unknown mixed classesin

both train and test phase. However, we constructed an Harsh learning condition in the experiment because the overall percentage of genuine data is only 16.7%, which is almost impossible in practice. But it can still detect some HTs without golden reference, which also makes it a competitive alternative in application.

5 Conclusion

Detection of stealthy Hardware Trojans violating runtime operations is significantly challenging. In this paper, we propose a ML involved framework to classify the genuine and Trojan-insert circuit using characterized side-channel power in instruction-level. Various features including instruction and Trojan types are well-extracted to construct exclusive feature sets. Experimental results on Altera FPGA represented that distinct ML methods have different sensitivities to the features, which can greatly fluctuate the accuracy. Naïve Bayes reached the best average accuracy, and the SVM-2C consumed the shortest CPU running time. We also proved that the unsupervised one-class SVM can detect HTs without golden reference in the range of 17% to 72% even in Harsh condition.

In the future, the optimized classification ML methods can be inserted into chips after getting well-trained to predict unknown HTs in order to accomplish real runtime detection.

References

[1] Tehranipoor M, Koushanfar F. A survey of hardware Trojan taxonomy & detection [J]. IEEE Design & Test of Computers, 2010, 27(1): 10-25.

[2] Rad R, Plusquellic J, Tehranipoor M. Sensitivity analysis to hardware Trojans using power supply transient signals [C]// 2018 IEEE International Workshop on Hardware-Oriented Security and Trust. Anaheim, CA, USA: IEEE Press, 2008: 3-7.

[3] Hou B, He C H, Wang L W, et al. Hardware Trojan detection via current measurement: a method immune to process variation effects [C] // 2014 10th International Conference on Reliability, Maintainability and Safely (ICRMS). Guangzhou: IEEE Press, 2015: 1039-1042.

[4] Aarestad J, Acharyya D, Rad R, et al. Detecting Trojans through leakage current analysis using multiple supply pad I_{DDQs} [J]. IEEE Transactions on Information Forensics and Security, 2010, 5(4): 893-904.

[5] He J J, Zhao Y Q, Guo X L, et al. Hardware Trojan detection through chip-free electromagnetic side-channel statistical analysis [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017,25(10): 2939-2948.

[6] Koushanfar F, Potkonjak M. CAD-based security, cryptography, and digital rights management [C]//2007 44th ACM/IEEE Design Automation Conference. San Diego, CA, USA: IEEE Press, 2007: 268-269.

[7] Wei S, Potkonjak M. Scalable consistency-based hardware Trojan detection and diagnosis [C] // 2011 5th IEEE International Conference on Network and System Security. Milan, Italy: IEEE Press, 2011: 176-183.

[8] Liu Y, Jin Y E, Nosratinia A, et al. Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017, 25(4): 1506-1519.

[9] Karimi N, Kanuparthi A K, Wang X Y, et al. MAGIC: malicious aging in circuits/cores [J]. ACM Transactions on Architecture and Code Optimization, 2015, 12(1): 1-25.

[10] Wang X X, Tehranipoor M, Plusquellic J. Detecting malicious inclusions in secure hardware: challenges and solutions [C] // 2008 IEEE International Workshop on Hardware-Oriented Security and Trust. Anaheim, CA, USA: IEEE Press, 2008: 15-19.

[11] Chakraborty R S, Wolff F, Paul S, et al. MERO: A statistical approach for hardware Trojan detection [C]//11th International Workshop on Cryptographic Hardware and Embedded Systems. Lansanne, Switzerland: Springer, 2009: 396-410.

[12] Rai D, Lach J. Performance of delay-based Trojan detection techniques under parameter variations [C] // 2009 IEEE International Workshop on Hardware-Oriented Security and Trust. San Francisco, CA, USA: IEEE Press, 2009: 58-65.

[13] Li X, Wang X, Zhang Y, et al. Hardware trojan detection method based on multiple side-channels analysis [J]. Computer Simulation, 2015, 32(3): 216-219. (in Chinese).

[14] Narasimhan S, Du D D, Chakraborty R S, et al. Hardware trojan detection by multiple-parameter side-channel analysis [J]. IEEE Transactions on Computers, 2013, 62(11): 2183-2195.

[15] Salmani H, Tehranipoor M, Plusquellic J. A layout-aware approach for improving localized switching to detect hardware Trojans in integrated circuits [C]//2010 IEEE International Workshop on Information Forensics and Security. Seattle, WA, USA: IEEE Press, 2010: 1-6.

[16] Forte D, Bao C X, Srivastava A. Temperature tracking: An innovative run-time approach for hardware Trojan detection [C] // 2013 IEEEE/ACM International Conference on

Computer-Aided Design(ICCAD). San Jose,CA,USA: IEEE Press, 2013: 532-539.

[17] Zhao H, Kwiat K, Kamhoua C, et al. Applying chaos theory for runtime Hardware Trojan detection [C]// 2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). Verona, NY, USA: IEEE Press, 2015: 1-6.

[18] Jap D, He W, Bhasin S. Supervised and unsupervised machine learning for side-channel based Trojan detection[C]// 2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP). London,UK: IEEE Press, 2016: 17-24.

[19] Bao C X, Forte D, Srivastava A. On reverse engineering-based hardware Trojan detection[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 35(1): 49-57.

[20] Lodhi F K, Abbasi I, Khalid F, et al. A self-learning framework to detect the intruded integrated circuits[C]//2016 IEEE International Symposium on Circuits and System (ISCAS). Montreal,QC,Canada: IEEE Press, 2016: 1702-1705.

[21] Xue M F, Wang J, Hux A Q. An enhanced classification-based golden chips-free hardware Trojan detection technique[C]//2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST). Yilan,Taiwan,China: IEEE Press, 2016: 1-6.

[22] Inoue T, Hasegawa K, Yanagisawa M, et al. Designing hardware Trojans and their detection based on a SVM-based approach[C]//2017 IEEE 12th International Conference on ASIC (ASICON). Guiyang,China: IEEE Press, 2017: 811-814.

[23] Lodhi F K, Hasan S R, Hasan O, et al. Power profiling of microcontroller's instruction set for runtime hardware Trojans detection without golden circuit models [C]// Design, Automation & Test in Europe Conference & Exhibition (DATE),2017. Lausanne,Switzerland: IEEE Press, 2017: 294-297.

[24] Mazidi M A, Mazidi J G, Mckinlay R D. The 8051 microcontroller and embedded systems using assembly and C [M]. 2nd ed. New Jersey: PearsonEducation, 2007.

[25] Tehranipoor M, Salamani H. trust-HUB. [CP/OL]. (2006-03-06)[2019-11-11]. *https://www.trust-hub.org/*.

[26] Chang C C, Lin C J. LIBSVM: A library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3): 1-27.