Detection of Illegitimate Access to JTAG via Statistical Learning in Chip

Xuanle Ren[1,2], Vítor Grade Tavares[2], and R. D. (Shawn) Blanton[1]

[1]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
[2]Faculty of Engineering, University of Porto, Porto, Portugal

***Abstract*—IEEE 1149.1, commonly known as the joint test action group (JTAG), is the standard for the test access port and the boundary-scan architecture. The JTAG is primarily utilized at the time of the integrated circuit (IC) manufacture but also in the field, giving access to internal sub-systems of the IC, or for failure analysis and debugging. Because the JTAG needs to be left intact and operational for use, it inevitably provides a "backdoor" that can be exploited to undermine the security of the chip. Potential attackers can then use the JTAG to dump critical data or reverse engineer IP cores, for example. Since an attacker will use the JTAG differently from a legitimate user, it is possible to detect the difference using machine-learning algorithms. A JTAG protection scheme, SLIC-J, is proposed to monitor user behavior and detect illegitimate accesses to the JTAG. Specifically, JTAG access is characterized using a set of specifically-defined features, and then an on-chip classifier is used to predict whether the user is legitimate or not. To validate the effectiveness of the approach, both legitimate and illegitimate JTAG accesses are simulated using the OpenSPARC T2 benchmark. The results show that the detection accuracy is 99.2%, and the escape rate is 0.8%.**

I. Introduction

IEEE 1149.1, commonly known as the joint test action group (JTAG), is the standard for the test access port and the boundary-scan architecture. JTAG is widely used in modern chips because it dramatically improves testability and expedites the testing process. It was estimated that the testing of a complex board is at least ten times faster due to the use of the JTAG [1]. JTAG is also used in the field [2]. It has become the primary interface for chip debugging and firmware programming [3].

High testability provided by the JTAG also inevitably undermines chip security. Specifically, the JTAG defines undocumented functions, namely backdoors, that can be exploited for illegitimate use. First, scan chains provide an interface to access on-chip data. For instance, cryptographic keys (e.g., DES and RSA [4]) from ICs can be derived by shifting-out the data from the internal scan chain and analyzing the relevant bits within the chain [5], [6]. Second, many other user-defined functions of the JTAG are also undocumented, rendering the chip more susceptible to attacks especially due to the fact that reverse engineering techniques have become powerful [3], [7]–[11]. These user-defined functions, designed for debugging and programming, can control and observe many on-chip resources. Once attackers have gained access, they can read on-chip data or even modify the chip operation. For instance, programmable logic controller (PLC) firmware, which provides a software-driven interface between system inputs and physical outputs, has been shown to be easily extractable and modifiable using the JTAG [8].

Various schemes have been proposed to protect the JTAG [12]–[18]. One technique involves disabling the JTAG before the device is sold to the consumer; however this disables in-field testing and debugging [12]. Access to the JTAG can also be obfuscated by distributing its ports across the chip or board, but the ports can be easily located using an approach described in detail in Section V. On-chip compression/compaction, which is originally aimed at expediting manufacturing test, also protects the JTAG. It obfuscates test responses, preventing them from being directly observed by an attacker [13], but it does not obfuscate user-defined backdoors [14]. Another scheme involves password-based authentication which requires a correct password to access the JTAG [15]–[17]. However, it modifies IEEE 1149.1 due to the support for lock/unlock commands. Another risk is password leakage, especially when all fabricated instances share the same password. Protocol-based schemes use more complex techniques for authentication [18]–[21]. Specifically, a trusted server is used to manage the multi-stage authentication between the user and the device, but has the drawback of requiring network availability. The aforementioned protection schemes mainly target scan chains, while protecting user-defined backdoors has gained little attention. However, as chips become more complex and have more debugging functions, these backdoors have introduced more vulnerabilities that need to be addressed [7]–[11].

In this paper, a JTAG protection scheme using statistical learning in chip (SLIC-J) is proposed to monitor user behavior and detect illegitimate access to the JTAG, ultimately protecting user-defined backdoors from being attacked [22], [23]. A similar idea was also proposed in [24]–[26], where an on-chip, neural-network classifier is employed for detecting Trojans in real time. SLIC-J is based on the fact that illegitimate users are prone to behave differently than legitimate users because they are not aware of which JTAG functions are implemented and how they should be operated. It is assumed that the attacker can access the JTAG ports of real chips using tools, such as JTAGulator and ftjrev [27], [28], and their goal is to discover the undocumented JTAG functions. SLIC-J characterizes JTAG access using a set of specifically-defined features. An on-chip classifier then predicts whether the user is legitimate or not.

978-3-9815370-4-8/DATE15/©2015 EDAA

This work has three main contributions:

- It is, as far as authors know, the first work that uses learning-based approach to ensure JTAG security. It is compliant with IEEE 1149.1, and does not compromise testability. It is password-free, and thus avoids the risk of password theft or eavesdropping.
- Various JTAG functions of the OpenSPARC T2 are described and existing strategies for attacking the JTAG are summarized.
- SLIC-J is demonstrated to have high detection accuracy in Section V, rendering illegitimate access to the JTAG even more difficult.

The rest of this paper is organized as follows. In Section II, the state-of-the-art of JTAG attacks, as well as JTAG functions, are reviewed. Section III elaborates upon the scheme of SLIC-J, and Section IV describes the hardware implementation. The simulation results are presented in Section V. Section VI discusses the limitations of SLIC-J. Finally, Section VII concludes the paper.

II. JTAG Attacks

In this section, JTAG functions as well as their vulnerabilities with respect to various attacks are reviewed. In Table 1, OpenSPARC T2 is used as an example to demonstrate typical JTAG functions. These functions are defined for debugging and programming, but attackers can also use them to access the chip. Table 1 also indicates that each JTAG function is achieved by a set of instructions that have successive opcodes.

JTAG functions	Opcode	No. of instr.
Control register configuration	0x08 - 0x10	9
Memory BIST	0x13 - 0x1B	9
Direct memory observe	0x1C - 0x1E	3
Electronic fuse	0x28 - 0x2F	8
Shadow scan	0x30 - 0x38	9
Clock control	0x40 - 0x46	6
Debug control register	0x48 - 0x52	10
L2 cache R/W	0x58 - 0x5B	4
Logic BIST	0x60 - 0x65	6
Internal scan	0x80 - 0x84	5

Table 1: The user-defined JTAG functions of the OpenSPARC T2 used for debugging and programming.

JTAG attacking strategies are summarized as three major steps and nine sub-steps (Table 2) [7]–[11]. First, the JTAG ports are identified physically. Second, the profile of each data register (DR) is explored. Specifically, sub-step 5 checks if each DR can be captured or updated. A DR that can be captured may extract data out of the chip, while a DR that can be updated may store data into the chip. Sub-step 6 identifies the internal scan chains based on the belief that they should be extremely long. Third, the undocumented JTAG functions are investigated. Specifically, sub-step 7 separates opcodes into bundles based on the belief that opcodes belonging to the same function are adjacent to each other. Sub-steps 8 and 9 determine the nature of each DR and investigate how opcodes collaborate to complete a JTAG function. For example, a 40-bit DR may represent an address, and a 64-bit DR may represent the data read by the address.

Steps	Sub-steps	
Identify the JTAG ports physically	1	Identify the location of the JTAG ports on chip/board
Find the basic profile of data registers	2	Identify the length of the instruction register (IR)
	3	Identify the length of each date register (DR)
	4	Find the opcodes that are not associated with any DR
	5	Check if each DR can be captured/updated
	6	Identify the internal scan chains
Investigate the undocumented functions	7	Separate opcodes into bundles based on their associated functions
	8	Determine the nature (control versus data) of each DR
	9	Investigate adjacent opcodes for characterizing any interactions between them

Table 2: Attacking strategies are organized as three major steps and nine sub-steps.

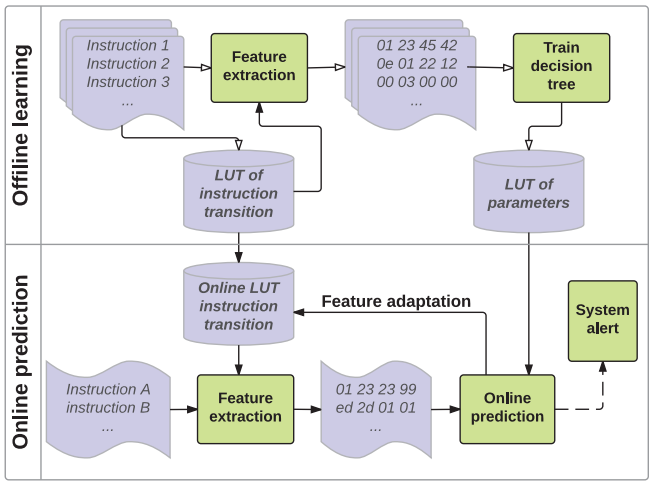


Figure 1: SLIC-J consists of two phases: offline learning and online prediction.

III. Machine Learning

To detect the attacking strategies summarized in Section II, SLIC-J uses a two-phase scheme: offline learning and online prediction (Figure 1). In offline learning, features are extracted from JTAG programs for training a decision-tree classifier that is stored in a non-volatile memory as a set of "if-then" rules. For online prediction, the same features are extracted and supplied to the decision-tree classifier for making a prediction within dozens of clock cycles.

A. Feature extraction

JTAG access is characterized using a set of specifically-defined features (Table 3). The features are collected during the execution of a JTAG instruction. Specifically, the features are captured when the instruction register (IR) is updated by a new opcode (Figure 2). The features characterize JTAG access

No.	Feature description
1	Higher 4 bits of the JTAG instruction
2	Lower 4 bits of the JTAG instruction
3	No. of clock cycles in shift-DR
4	No. of clock cycles in run-test/idle
5	No. of clock cycles in test-logic-reset
6	No. of test-mode-select (TMS) transitions
7	Legal opcode?
8	Normal transition?

Table 3: JTAG access is characterized using eight specifically-defined features.

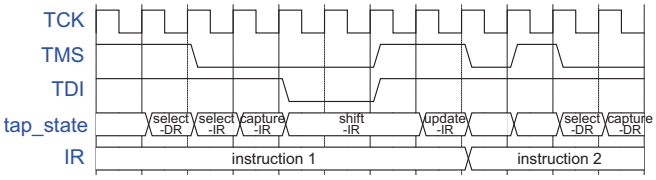


Figure 2: The features are collected during the execution of a JTAG instruction.

from two aspects: intra-instruction statistics (features 1-7), and inter-instruction transition (feature 8).

Feature 3 depicts the number of DR-shift cycles. If the attacker is not aware of the length of the selected DR, then there may be more or fewer shift cycles employed. Features 4 and 5 depict the number of clock cycles in the run-test/idle and the test-logic-reset states, respectively. Entering the test-logic-reset state may clear the IR and some DRs, which needs to be identified. Feature 6 is the number of the test-mode-select (TMS) transitions, i.e., from 0 to 1, or from 1 to 0. It is relevant in two cases: 1) Reset-type instructions require fewer TMS transitions because they do not select any DR. 2) Pause-DR, which is useful when a shifting DRs has to be temporarily suspended, causes more TMS transitions. Feature 7 indicates whether the value loaded into the IR is a legal opcode that corresponds to one of the JTAG functions.

Feature 8 is a binary feature that indicates if the transition from one instruction to the next is normal (i.e., commonly used by legitimate users) based on a look-up table (LUT) built beforehand. Specifically, each JTAG instruction has a list of normal follow-on instructions stored in the LUT. If the next instruction is in the list, feature 8 is "1"; otherwise, it is "0".

B. Decision tree learning

The features described in Section III-A are fed into an on-chip decision-tree classifier. The learned tree can be represented as a set of "if-then" rules, each one associated with a tree node. The classification process begins from the root node, and descends to branches until it reaches a leaf node.

Decision tree training employs a top-down, greedy search through the space of possible trees [29]. To avoid overfitting, the data set is partitioned into two parts: one for training the full tree, and the other for validating the utility of post-pruning nodes. Subtrees are removed if the resulting pruned tree performs no worse than the original one over the validation set.

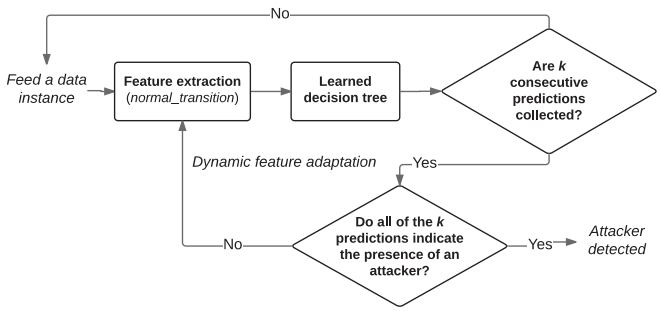


Figure 3: Feature 8 ($normal\_transition$) is adapted based on the predictions for every k consecutive JTAG instructions.

C. Dynamic feature adaptation

Due to the variance that naturally occurs within both legitimate and illegitimate JTAG accesses, the labeling of the user is delayed until sufficient evidence is collected as shown in Figure 3. Specifically, only when all consecutive predictions within a period indicate the presence of an attacker, the user is labeled as an attacker. Further, a dynamic feature adaptation strategy is proposed to improve detection accuracy.

In SLIC-J, $normal\_transition$, rather than the learned classifier, is adapted since it can significantly affect the prediction of the classifier. Before introducing the adaptation strategy, several parameters are defined. A period denotes k consecutive instructions, and m denotes the number of illegitimate predictions within a period. In addition, two thresholds T_h and T_l, subject to

$$0 \leq T_l \leq T_h \leq k$$

are also defined. The adaptation strategy is:

1) $m = k$, the user is very likely an attacker;

2) $T_h \leq m < k$, the user is likely an attacker, so set all instruction transitions within the period to be abnormal (i.e., removing the corresponding entries from the instruction-transition LUT described in Section III-A);

3) $T_l < m < T_h$, the user may or may not be an attacker, so take no action;

4) $m \leq T_l$, the user is likely to be legitimate, so set all instruction transitions within the period to be normal (i.e., adding the corresponding entries into the instruction-transition LUT).

An example is provided ($k = 4$, $T_h = 3$, and $T_l = 1$) to demonstrate how the dynamic feature adaptation works (Table 4). In period 1, three predictions are illegitimate, so the only normal transition (i.e., from instruction A to instruction B) is removed from the instruction-transition LUT. In period 2, two predictions are illegitimate, so no action is taken. It is noticed that the transition from instruction A to instruction B reports "0" rather than "1". In period 3, all predictions are illegitimate, in which case the system is alerted of the inappropriate use of the JTAG.

Opcode	Normal_transition	Prediction	
...	...	...	...
Instr. 1	0	Illegitimate	period 1
Instr. 2	0	Illegitimate	
Instr. A	0	Illegitimate	
Instr. B	**1**	Legitimate	
Instr. 3	1	Illegitimate	period 2
Instr. 4	1	Legitimate	
Instr. A	0	Legitimate	
Instr. B	**0**	Illegitimate	
Instr. 5	0	Illegitimate	period 3
Instr. 6	0	Illegitimate	
Instr. 7	0	Illegitimate	
Instr. 8	1	Illegitimate	
...	...	...	...

Table 4: This JTAG program shows how feature 8 ($normal\_transition$) is adapted using two thresholds, T_h and T_l. In this example, two heuristic values are used, i.e., T_h=3 and T_l=1. The transition from instruction A to instruction B is reported to be normal in period 1; however, it becomes abnormal in period 2 because the number of illegitimate predictions in period 1 is equal to T_h.

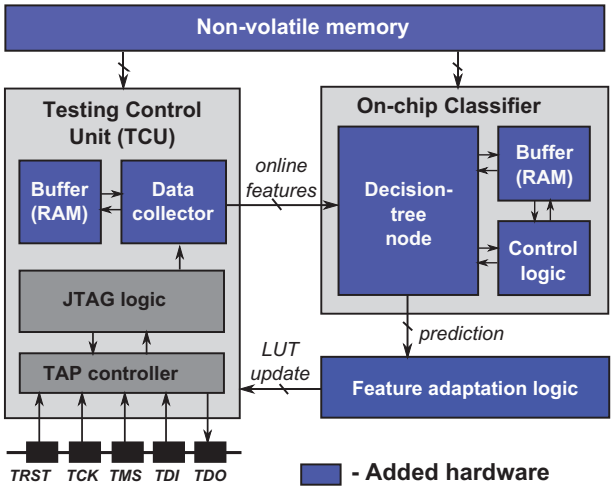


Figure 4: The architecture of SLIC-J within the Testing Control Unit (TCU) module in the OpenSPARC T2.

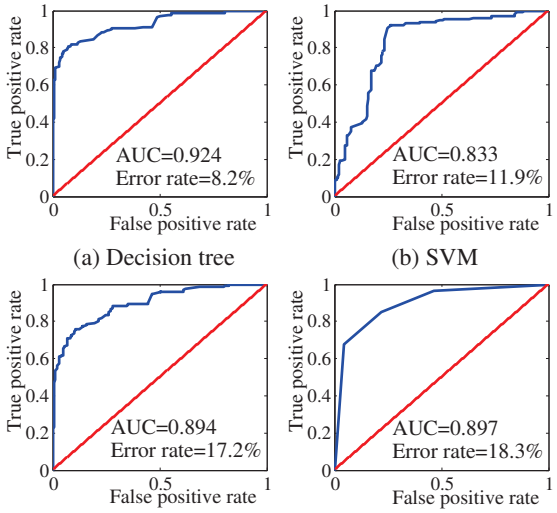
IV. Hardware Implementation

SLIC-J is implemented within the OpenSPARC T2 benchmark which is an open-source 64-bit eight-core multi-threaded microprocessor [30]. The JTAG of the OpenSPARC T2 is defined in the Testing Control Unit (TCU) module. As detailed in Section II, the JTAG controls several testing and debugging functions. Synopsys VCS [31] is used for behavior-level simulation, and Synopsys Design Compiler [32] is used for synthesizing the design.

As shown in Figure 4, SLIC-J contains four modules, namely, a data collector, a non-volatile memory, an on-chip classifier, and feature adaptation logic. When the chip is powered on, the instruction-transition LUT is loaded into the buffer for the data collector from the non-volatile memory; in addition, the "if-then" rules for the learned tree are loaded into the buffer for the classifier. The data collector captures all features periodically and sends them to the classifier. After the classifier receives the features, it initializes the classification, one node being processed per clock cycle. The architecture of the universal tree node proposed in [33] is used. Finally, after the prediction is made, the feature adaptation logic updates the instruction-transition LUT.

	TCU	TCU with SLIC-J
Area/μm^2	40520	74962
Timing/ns	7.46	9.17
Power/μW	299	353

Table 5: The synthesis result shows that SLIC-J adds comparable area to the TCU module (i.e., the original JTAG). Additionally, it is slower (19%) than the original JTAG.



(a) Decision tree (b) SVM

(c) Multilayer perceptron (d) KNN

Figure 5: The simulation results are given as Receiver Operating Characteristic (ROC) curves. The error rate indicates the percentage of misclassified instances, and the AUC (i.e., the area under the upper curve) indicates the capability of a classifier to make correct prediction for a randomly chosen instance. (a) Decision tree, with post-pruning. (b) SVM, with the radial basis function kernel. (c) Multilayer perceptron, with one hidden layer that has ten neurons. (d) KNN, with K=3.

For online use, SLIC-J should have comparable timing constraint to the TCU module, and consume as small area as possible. The synthesis result (Table 5) indicates that SLIC-J adds comparable area to the TCU module (i.e., the original JTAG). SLIC-J consumes 0.32% chip area of the OpenSPARC T2 design. In addition, the critical path is slower (19%) than the original JTAG, indicating that the architecture in [33] needs to be improved in order to be more compatible with the timing specification of the OpenSPARC T2.

V. Experiment

To validate the effectiveness of SLIC-J, both legitimate and illegitimate JTAG accesses are simulated using the OpenSPARC T2 benchmark. A variety of illegitimate accesses

are generated based on the attacking strategies described in Section II. Specifically, 66 illegitimate programs, containing 6,935 JTAG instructions, are generated. Another 44 legitimate programs, containing 7,189 JTAG instructions, are generated to mimic legitimate JTAG accesses based on the OpenSPARC T2 documentation. For every individual instruction, SLIC-J collects the features and makes a prediction as described in Section III. Both legitimate and illegitimate instances are partitioned into two parts: one for training and the other for testing.

Several commonly used machine-learning algorithms, including a decision tree, support vector machine (SVM), multilayer perceptron, and *K*-nearest-neighbor (KNN), are investigated. For the decision tree, the "training" set is partitioned into two parts: one for training the full tree and the other for eliminating overfitting. For SVM, several kernel functions, including linear, quadratic, polynomial and the radial basis function, are investigated, and the radial basis function kernel achieves the highest accuracy. For multilayer perceptron, it is a neural network that has one hidden layer with ten neurons. For KNN, the optimized value for *K* found through simulation is three.

The Receiver Operating Characteristic (ROC) curves [34] are plotted to compare their performance (Figure 5). The area under the curve (AUC), equal to the probability that a classifier ranks a randomly chosen illegitimate instance higher than a randomly chosen legitimate one, measures the capability of a classifier to make the correct prediction for a randomly chosen instance. In Figure 5, the AUC is the area under the upper curve, while the lower line represents a random classifier. The results indicate that the accuracy of the decision tree, 91.8%, surpasses all other algorithms. The AUC of the decision tree, 0.924, is also the best. In addition, the results also demonstrate that the detection of illegitimate JTAG accesses is not an easy-to-classify problem since some powerful classifiers (e.g., SVM and multilayer perceptron) do not perform well.

Finally, a program, comprising interleaved legitimate and illegitimate JTAG accesses, is created. In addition, the delayed labeling and the dynamic feature adaptation described in Section III-C are used, i.e., the user is labeled legitimate or illegitimate every four consecutive instructions, and $normal\_transition$ is adapted (Table 6). The attacking strategies described in Section II are simulated, and the performance is assessed by detection accuracy (the fraction of correct labelings out of all labelings) and escape rate (the fraction of attacks that escape detection out of all attacks). By using the dynamic feature adaptation, the overall detection accuracy is improved from 97.9% to 99.2%, and the overall escape rate is reduced from 1.4% to 0.8%. The escape rate indicates that an attacker can escape detection at the probability of 0.8%.

VI. Discussion

Although the experiment results of Section V are promising, they can only be considered valid under certain assumptions with respect to the attacker. In this section, several scenarios are described that would circumvent the effectiveness of SLIC-J. More, the ability of other approaches to effectively cope with these scenarios is discussed, motivating the need for a combination of techniques for protecting the JTAG.

As mentioned, there are several attacker scenarios that circumvent the protection provided by SLIC-J. First, if an attacker is aware of which JTAG functions are implemented and how they should be operated, then SLIC-J has no effect. Second, if an attacker can perform attacks on multiple chips, then the attacker is more likely to escape detection by SLIC-J. Specifically, if an attacker is detected and prevented from accessing the JTAG, the attacker can turn to another chip as if the chip has been reset. Third, if an attacker knows how SLIC-J works (i.e., the decision-tree algorithm, the length of a labeling period, and the use of the dynamic feature adaptation), the attacker may hide illegitimate accesses within legitimate ones to escape detection by SLIC-J. In order to mitigate this type of attack, the length of a labeling period k should not be too large. Fourth, SLIC-J cannot guarantee that "previously-unseen" behavior can be detected. For example, if an attacker develops a novel attacking strategy that can bypass the extracted features, then it may be possible that the attacker can escape detection by SLIC-J.

Table 7 compares SLIC-J with other JTAG protection schemes. It is difficult however to compare their performance directly because SLIC-J targets the undocumented functions rather than all functions. One advantage however is that SLIC-J and the other schemes are orthogonal, meaning that they can be combined to achieve complementing protection for the JTAG. For example, if SLIC-J is augmented with the password-based authentication, then the attacker has to explore both the password and how to operate the JTAG legitimately.

VII. Conclusion

Security is becoming a central problem for modern integrated systems. The JTAG, designed as the testing interface, exposes chips to security risks due to the observability and controllability it enables. In this paper, a JTAG protection scheme, SLIC-J, is proposed to detect illegitimate access to the JTAG. The performance is measured over a variety of attacking strategies. Experiment results demonstrate a hacker-detection accuracy of 99.2%, and an escape rate of 0.8%. These experiments show that SLIC-J can effectively protect the undocumented functions of the JTAG from being attacked given that 1) the attacker has no prior knowledge of which JTAG functions are implemented and how they should be operated, 2) the attacker does not have a large number of chips to repeat the attacks, 3) the length of a labeling period, *k*, is properly set, and 4) the features defined by SLIC-J are not bypassed by the attacker.

Acknowledgment

This work is supported by FCT, Portugal, under Grant BD/28163/2006 (project ref. CMU-PT/SIA/0005/2009), and by the European Regional Development Fund (ERDF) through

JTAG attacking strategies	With dynamic feature adaptation?	TP	TN	FP	FN	Detection accuracy	Escape rate
Identify the JTAG ports physically	No	23	26	2	0	96.1%	0
	Yes	24	26	1	0	98.0%	0
Find the basic profile of data registers	No	315	272	9	6	97.5%	1.9%
	Yes	322	276	2	2	99.3%	0.6%
Investigate the undocumented functions	No	551	442	10	7	98.3%	1.3%
	Yes	551	450	4	5	99.1%	0.9%

Table 6: The detection accuracy and escape rate for labeling a user as legitimate or illegitimate, where TP=true positive, TN=true negative, FP=false positive, FN=false negative, Detection accuracy=(TP+TN)/(TP+TN+FP+FN), and Escape rate=FN/(TP+FN).

JTAG protection schemes	Protected target	Is the attacker aware of how to operate JTAG?	Hardware overhead	Potential risks	Security level
Disable JTAG	All JTAG functions	Maybe	Small	Invasive attack	High
On-chip compression/compaction	Boundary scan	Maybe	Medium	Differential attack	Low
Password-based authentication	All JTAG functions	Maybe	Medium	Password leakage	Medium
Protocol-based authentication	All JTAG functions	Maybe	Large	Eavesdropping, network fails	High
SLIC-J	Undocumented JTAG functions	No	Medium	The attacker has prior knowledge	High

Table 7: The comparison between SLIC-J and other JTAG protection schemes.

the Programme COMPETE (operational programme for competitiveness). This work is also supported by ICTI, a collaboration between the government of Portugal and Carnegie Mellon University.

REFERENCES

[1] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer, 2000.

[2] J. Da Rolt, A. Das, G. Di Natale, M. Flottes, B. Rouzeyre, and I. Verbauwhede, "Test versus security: past and present," *Emerging Topics in Computing*, vol. 2, no. 1, pp. 50–62, 2014.

[3] "Design security in nonvolatile flash and antifuse FPGAs," Actel, Tech. Rep., 2002.

[4] B. Yang, K. Wu, and R. Karri, "Scan-based side-channel attack on dedicated hardware implementations of data encryption standard," in *International Test Conference*, 2004.

[5] Y. Liu, K. Wu, and R. Karri, "Scan-based attacks on linear feedback shift register based stream ciphers," *Design Automation of Electronic Systems*, vol. 16, no. 2, pp. 1–15, 2011.

[6] S. S. Ali, O. Sinanoglu, S. M. Saeed, and R. Karri, "New scan-based attack using only the test mode," in *International Conference on Very Large Scale Integration*, no. 1, 2013.

[7] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.

[8] I. Breeuwsma, "Forensic imaging of embedded systems using JTAG (boundary-scan)," *Digital Investigation*, vol. 3, no. 1, pp. 32–42, 2006.

[9] F. Domke, "Blackbox JTAG reverse engineering," Tech. Rep., 2009.

[10] S. Skorobogatov and C. Woods, *Breakthrough silicon scanning discovers backdoor in military chip*. Springer, 2012.

[11] I. Slochinsky, "Introduction to embedded reverse engineering for PC reversers," in *REcon Conference*, 2010.

[12] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "Thwarting scan-based attacks on secure-ICs with on-chip comparison," *Very Large Scale Integration Systems*, vol. 22, no. 4, pp. 947–951, 2014.

[13] "High quality test solutions for secure applications," Mentor Graphics, Tech. Rep., 2010.

[14] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "Are advanced DfT structures sufficient for preventing scan-attacks?" in *VLSI Test Symposium*, 2012.

[15] A. Das and U. Kocaba, "PUF-based secure test wrapper design for cryptographic SoC testing," in *Design, Automation and Test in Europe*, 2012.

[16] S. Paul, "Vim-scan: A low overhead scan design approach for protection of secret key in scan-based secure chips," in *VLSI Test Symposium*, 2007.

[17] F. Novak and A. Biasizzo, "Security extension for IEEE std 1149.1," *Journal of Electronic Testing*, vol. 22, no. 3, pp. 301–303, 2006.

[18] J. Dworak and A. Crouch, "Don't forget to lock your SIB: Hiding instruments using P16871," in *International Test Conference*, 2013.

[19] B. Buskey and B. Frosik, "Protected JTAG," in *International Conference on Parallel Processing Workshops*, 2006.

[20] A. Das, J. Da Rolt, S. Ghosh, and S. Seys, "Secure JTAG implementation using schnorr protocol," *Electronic Testing*, vol. 29, no. 2, pp. 193–209, 2013.

[21] K.-Y. Park, S.-G. Yoo, and J.-H. Kim, "Debug port protection mechanism for secure embedded devices," *Semiconductor Technology and Science*, vol. 12, no. 2, pp. 240–253, 2012.

[22] R. D. Blanton, X. Li, K. Mai, D. Marculescu, R. Marculescu, J. Paramesh, J. Schneider, and D. Thomas, "SLIC: statistical learning in chip," in *International Symposium on Integrated Circuits*, 2014.

[23] D. Ricketts, J. A. Bain, Y. Luo, R. D. Blanton, K. Mai, and G. K. Fedder, "Enhancing CMOS using nanoelectronic devices, a perspective on hybrid integrated systems," in *Proceedings of the IEEE*, 2010.

[24] Y. Jin, D. Maliuk, and Y. Makris, "Post-deployment trust evaluation in wireless cryptographic ICs," in *Design, Automation and Test in Europe*, 2012.

[25] Y. Jin and D. Sullivan, "Real-time trust evaluation in integrated circuits," in *Design, Automation and Test in Europe*, 2014.

[26] D. Maliuk, H.-G. Stratigopoulos, H. Huang, and Y. Makris, "Analog neural network design for RF built-in self-test," in *International Test Conference*, 2010.

[27] "JTAGulator," http://www.grandideastudio.com/portfolio/jtagulator/.

[28] "ftjrev," http://www.alexforencich.com/wiki/en/projects/ftjrev/start/.

[29] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[30] "OpenSPARC T2," http://www.oracle.com/technetwork/systems/opensparc/opensparc-t2-page-1446157.html, Oracle.

[31] "VCS," http://www.synopsys.com/Tools/Verification/ FunctionalVerification/Pages/VCS.aspx, Synopsys.

[32] "Design Compiler," http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages/default.aspx, Synopsys.

[33] J. Struharik, "Implementing decision trees in hardware," in *International Symposium on Intelligent Systems and Informatics*, 2011.

[34] "ROC curve," http://www.mathworks.com/help/stats/perfcurve.html.