A Game-Theoretic Approach for Testing for Hardware Trojans

Charles A. Kamhoua, *Senior Member, IEEE*, Hong Zhao, *Senior Member, IEEE*, Manuel Rodriguez, and Kevin A. Kwiat

Abstract—The microcircuit industry is witnessing a massive outsourcing of the fabrication of ICs (Integrated Circuit), as well as the use of third party IP (Intellectual Property) and COTS (Commercial Off-The-Shelf) tools during IC design. These issues raise new security challenges and threats. In particular, it brings up multiple opportunities for the insertion of malicious logic, commonly referred to as a hardware Trojan, in the IC. Testing is typically used along the IC development lifecycle to verify the functional correctness of a given chip. However, the complexity of modern ICs, together with resource and time limitations, makes exhaustive testing commonly unfeasible. In this paper, we propose a game-theoretic approach for testing digital circuits that takes into account the decision-making process of intelligent attackers responsible for the infection of ICs with hardware Trojans. Testing for hardware Trojans is modeled as a zero-sum game between malicious manufacturers or designers (i.e., the attacker) who want to insert Trojans, and testers (i.e., the defender) whose goal is to detect the Trojans. The game results in multiple possible mixed strategy Nash equilibria that allow to identify optimum test sets that increase the probability of detecting and defeating hardware Trojans in digital logic. Results also show that the minimum number of Trojan classes tested by the defender and the fines imposed to the attacker can deter rational as well as irrational attackers from infecting circuits with Trojans.

Index Terms—Hardware Trojan, cyber security, game theory, functional testing, integrated circuit

1 Introduction

A hardware Trojan is a malicious modification of the circuitry of an Integrated Circuit (IC) [1], [2], [3]. A hardware Trojan is inserted into a main circuit at manufacturing or during design, and is mostly inactive until it is activated by a rare condition. When activated, it produces an error in the circuit, potentially leading to catastrophic consequences. The threat of serious, malicious IC alterations is of special concern to government agencies, military, finance, energy and political sectors.

The threat of hardware Trojans has become more pronounced due to today's massive outsourcing of the IC manufacturing processes (*fab-less model*), as well as from the increased reliance on hardware COTS (Commercial Off-The-Shelf) components. A good example of the latter are legacy military systems including aerospace and defense platforms, which are facing obsolescence due to their extended lifetime (e.g., often because of budgetary decisions) and which rely on the use of COTS for the maintenance and replacement of their electronics. Most chip designers have now gone *fab-less*, outsourcing their manufacturing to offshore foundries. In doing so, they avoid the huge expense of building a state-of-the-art fab. Trust for Trojan-free fabrication of the chip is often placed upon foundries overseas. This gives many possibilities for potential attackers to maliciously alter the IC circuitry and insert hardware Trojans.

Hardware Trojans are designed to be *stealthy*. They can be very small (e.g., consisting of only few transistors) so that they elude physical inspections and side-channel analyses of chips. In addition, they are commonly triggered by rare conditions so that they pass functional tests. Even if a Trojan is physically very small, it can still have a severe impact, ranging from disabling or destroying the chip to enabling backdoors or transmitting confidential information.

The major techniques employed to detect hardware Trojans are *physical inspection*, *side-channel analysis*, *runtime mechanisms*, and *logic testing* [4]. *Physical inspection* (also referred to as *destructive reverse engineering* or *optical reverse engineering*) consists of the analysis of the manufactu-red chip by reverse engineering (de-packaging, de-metallization, micro-photography). *Side-channel analysis* aims at measuring and analyzing a device's side-channel parameters such as supply current or path delay, to reveal unintended modifications to the circuit. *Runtime mechanisms* are meant to detect hardware Trojans at runtime while the system is in operation, typically using techniques relying on the online monitoring of critical operations. *Logic testing* (as applied to Trojan detection) aims at developing test patterns that can activate Trojans and propagate their effects to the circuit's output. An output's value is usually compared with the correct value as defined by the circuit's functional specifications so that a Trojan detection occurs if both values differ. Testing a circuit this way is thus a form of functional testing. In this paper, the focus is on logic (or functional) testing techniques of ICs.

Testing techniques usually assume a specific fault model and generate test cases aimed at uncovering faults pertaining to such model. For the testing of ICs, fault models are

- *C.A. Kamhoua, M. Rodriguez, and K.A. Kwiat are with the Air Force Research Laboratory, Information Directorate, Cyber Assurance Branch, Rome, NY 13441. E-mail: {charles.kamhoua.1, manuel.rodriguez-moreno.1.ctr, kevin.kwiat}@us.af.mil.*
- *H. Zhao is with the School of Computer Sciences and Engineering, Fairleigh Dickinson University, Teaneck, NJ 07666. E-mail: zhao@fdu.edu.*

Manuscript received 8 Oct. 2015; revised 30 Mar. 2016; accepted 3 May 2016. Date of publication 9 May 2016; date of current version 19 Oct. 2016.
Recommended for acceptance by S. Hu, Y. Jin, M. M. Tehranipoor, and K. Heffner.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMSCS.2016.2564963

commonly based on naturally occurring types of faults such as stuck-at faults, signal delays or transistor faults, as well as on accidental errors such as circuit's design flaws. Such faults are known as *random* in contrast to hardware Trojans, which are considered to be *intelligent* faults. Indeed, an attacker will insert Trojans within a circuit in a way that they are difficult to detect. For example, the attacker may design a Trojan triggered by rarely exercised circuit signals which are unlikely to be tested (e.g., due to low observability or controllability). The attacker will also design Trojans that cannot be detected by *known* tests (e.g., publicly available tests, standard tests, etc.).

To efficiently detect hardware Trojans, testing techniques need to be specifically designed to target intelligent faults. While a number of testing approaches exist for hardware Trojans (see Section 2), they do not explicitly address the decision-making process of intelligent attackers responsible for infecting circuits with Trojans. When there are two or more rational entities that face interdependent choices, we can use game theory to model their behavior and better understand the overall operating point of the system.

The main contribution of this paper is to propose the use of *game theory* to explicitly model the interactions between intelligent attackers and testers of digital circuits. As explained in Section 2, our work represents the first application of a game-theoretic approach to the field of hardware Trojan detection [5]. The proposed game-theoretic methodology helps identify optimum test sets that increase the probability of uncovering hardware Trojans. It also allows finding the optimum test conditions that can be used to force an attacker not to insert any Trojan so as to avoid severe penalties. In this respect, we formulate a non-cooperative game between malicious designers or manufacturers who seek to insert Trojans, and testers (who act as defenders) whose goal is to detect the Trojans. To solve this game, we analyze the Nash equilibrium point − representing the solution of the game—in which neither attacker nor defender has an incentive to change their strategy. The results show that the attacker is less likely to insert a high value (damage) Trojan because such high values Trojans are frequently tested by the defender. Moreover, a rational attacker is better off not to insert any hardware Trojans when the defender follows our testing procedure. Finally, the results also show that our testing strategy is robust against irrational attackers.

The paper is organized as follows. Section 2 describes the background and the related work. Section 3 provides an overview of the proposed testing approach for the detection of hardware Trojans. The core of the approach is the game theoretic model, which is introduced in Section 4. Section 5 illustrates the methodology via numerical analysis of the game theoretic model. In Section 6, our proposed game theoretic approach is applied to a real world example IC. Finally, Section 7 concludes the paper.

2 Background and Related Work

Functional testing of a digital system aims at validating the correct operation of the system with respect to its functional specification [6]. It commonly consists of generating inputs to the system and comparing the obtained outputs against a so-called *golden reference*. Testing of Integrated Circuits (IC) is usually carried out during the IC development cycle via *Automatic Test Pattern Generation* (ATPG). However, because of the stealthy nature of hardware Trojans, standard functional or ATPG testing is usually rendered as insufficient for detecting hardware Trojans. This is underscored in [7] by showing a test set for a simple digital circuit that detects all stuck-at-zero and stuck-at-one faults yet fails to detect the hardware Trojan. The specific application of functional testing to the detection of hardware Trojans has led to a number of approaches in the literature [7], [8], [9], [10], [11]. Some of them are described hereafter.

Wolff et al. [7] propose an approach to generate ATPG test vectors oriented towards the detection of hardware Trojans. Test vectors are optimized for functional Trojans based on combinational logic whose inputs and outputs are commonly attached to rarely triggered circuit nodes with low observability and controllability. Accordingly, the authors generate test vectors that exercise rarely activated parts of a circuit with the goal of triggering a Trojan and propagating its effects to the circuit output.

The authors in [8] introduced a probabilistic approach to detect Trojans in combinational circuits. It consists of a randomization technique to generate a unique signature from the circuit design, and a hypothesis testing technique to determine the presence of Trojans in the manufactured circuit. The signature is based on a probability distribution on the inputs such that the probability distribution of the outputs is unique for every functionally distinct circuit. Such an input distribution is used to generate test vectors applied to both the original design and the manufactured chip, and compare the outputs. If no differences are found, it is inferred with a certain confidence level that the fabricated chip is Trojan-free. Otherwise, a Trojan is considered to be present.

Chakraborty et al. [9] presented a testing approach to detect Trojans based on generating test patterns that trigger rarely exercised nodes of a circuit multiple times and on improving test length and detection coverage with respect to random and ATPG test patterns. The approach is applied to combinational logic based Trojans, and to a specific type of sequential Trojan (consisting of a clocked n-bit counter).

Hicks et al. [10] developed the Unused Circuit Identification (UCI) technique as part of their BlueChip implementation. UCI identifies parts of a hardware design that are not used or activated by any of the design verification tests, i.e., parts that could just be replaced by a single "wire". To do so, UCI relies on an algorithm that builds a data-flow graph of dependent signals, and then simulates the hardware design code using the verification tests to find signal pairs having the same value. The logic between those pairs is unused and considered suspicious, and replaced by trap-triggered emulated software in the actual system. Later on, the authors in [11] (comprising some of the original authors of UCI) found a subset of circuits capable of defeating UCI.

In addition to the above methodologies, *Design for Testability* (DFT) techniques facilitate the testing of digital logic [12]. They can provide the hardware means to support and enhance testing, such as observability and controllability mechanisms. DFT techniques have been used by a number of authors to specifically support the testing of hardware Trojans [13], [14], [15], [16]. Some examples are provided hereafter.

Chakraborty and Bhunia [13] propose a technique based on obfuscated circuit modes that helps prevent an attacker from inserting hard-to-find Trojans. It makes it difficult to an adversary to find nodes with low observability and controllability which allow constructing Trojan trigger conditions and payloads. The adversary then designs and inserts a Trojan based on wrong controllability/observability nodes, so the probability of triggering and detecting the Trojan during testing becomes higher.

Salmani et al. [14] propose an approach to increase circuit activity and reduce activation time of functional Trojans by increasing the transition probability of rarely transitioning gates. Transition probabilities are modeled using geometric distribution and estimated on the basis of the number of clock cycles needed to generate a transition on a net. They insert dummy flip-flops to increase the transition probability of nets when it is lower than a specific probability threshold.

Chakraborty et al. [15] aim at creating testable sub-circuits by adding extra inputs and outputs for signatures. Each module of a multi-module design defines a so-called transparent mode that executes self-testing circuitry designed to test rare events, and outputs a signature by combining its self-test and the input signature of a previous module. The hope is that Trojan tampering will affect the signature and reveal the Trojan. The authors claim that this method is useful against an attacker who has information about the functionality and logic structure of the IC.

Gassend et al. [16] propose chip structures to create PUFs (Physical Unclonable Functions). PUFs correspond to circuitry built on random uncontrollable parameters originated from process variations, meant to reliably and securely identify individual ICs.

Compared to the approaches cited above, our proposed testing methodology is built on a game-theoretic technique. To the best of our knowledge, this is the first work that adopts a game-theoretic approach to detect hardware Trojans [5]. Our approach, based on a mixed strategy Nash equilibrium, captures the defender's and attacker's possible best responses that are used to enhance testing for hardware Trojans. The use of game theory in our approach allows for selecting optimum sets of verification tests that maximize the chances of detecting a hardware Trojan inserted in a circuit by an intelligent attacker. Unlike many other Trojan testing approaches, our methodology is independent of the circuit type (either combinational or sequential) and the lifecycle development phase in which the testing happens (either IC design or manufacturing).

There has been some considerable work on applying game theory for security [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. For instance, zero-sum games are used in [17], [18] since the attacker's and defender's goals are purely conflicting. Most game theoretic models assume that all the players are rational [19], [20]. The research in [21], [22] relaxes the assumption of player's rationality and uses the mathematical framework of evolutionary game theory to model network security. Kamhoua et al. [22] use game theory to propose solutions to strategic cyber security situations in which an individual's success securing their asset depends on the trust of others to also secure their asset. In some scenarios, the security game is static [23], [24], but in others, the game model is repeated, or more generally stochastic [17], [19], [25]. A stochastic game is a generalization of a repeated game. In a repeated game, players play the same stage game in all periods, whereas in a stochastic game, the stage game can randomly change from one period to the next.

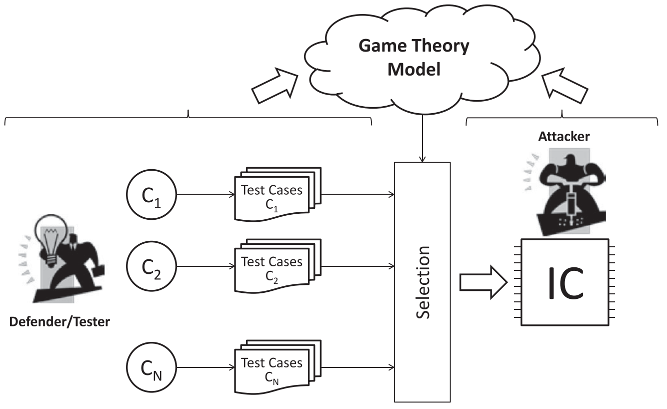


Fig. 1. Overview.

Cyber security games also consider what information each player knows. When the rules of the game, and each player's strategy and payoffs are assumed to be common knowledge, the cyber game is of complete information as in [23]. Otherwise, we have a game of incomplete information that can be formulated as a Bayesian game as in [24]. The work in [24], [26] modeled intrusion detection as a game. Information sharing in online social networks is modeled using a Markov decision process [27], [28] and a zero-sum Markov game in [18], [29]. The research in [25] uses a repeated voting game among replicated nodes to extend the mission survival times in a critical mission. Game theory has been applied to cyber survivability [25], [30], resilient control system design [20], and privacy [31]. The work in [32], [33] investigates replicas' diversity for cyber survivability in the framework of game theory. For an extended survey of game theory applied to cyber security, the reader is referred to [34], [35].

From our study of the literature, only the work by Kukreja et al. [36] applies game theory to the domain of *testing*. The authors model software test scheduling as a Stackelberg game between testers and developers. The testers act as defenders commit to a testing strategy. Developers play the role of attackers who may check-in insufficiently tested code to complete application functionality sooner. They compute a strong Stackelberg equilibrium that provides the optimal probability distribution of the test cases to be selected for a randomized schedule.

However, none of the approaches described above on game theory have addressed the problem of hardware Trojans. As far as we know, our work is the first application of game theory to the detection of hardware Trojans in digital circuits [5].

3 Trojan Detection Approach

Fig. 1 shows an overview of the proposed game theoretic approach for testing hardware Trojans. Modern ICs are extremely complex and it is unfeasible to develop test patterns that can verify the correctness of the entire chip or cover the entire logic space of the chip. Thus, a game-theoretic approach that identifies optimum sets of test patterns can greatly help improve a test's Trojan-detection efficiency. Such a game-theoretic model can take into account the

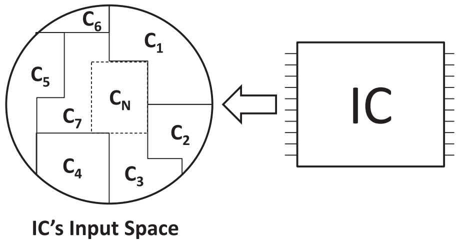


Fig. 2. Partition of the IC's input space.

decision-making process between an attacker and a tester, and allows the latter to select optimum sets of test cases developed out of specific test classes (namely: $C_1,\ C_2, \ldots,\ C_N$, as shown in Figs. 1 and 2).

We consider that the input space of the IC under test is divided into a partition of N arbitrary classes $C_1,\ C_2, \ldots,\ C_N$, as shown in Fig. 2. We do not set any specific constraints on the definition of each class, thus such a definition is entirely left to the tester.

For example, given a combinational circuit with q input pins, one could define one class per input combination, leading to a total number of classes N equal to 2^q. For large values of q, input combinations can be grouped depending on a goal value for N. In sequential circuits, the state machine of the IC can be taken into account to identify partition classes according to different criteria. For example, if the number of values (or types) of the IC output is well defined in the IC specifications, partition classes could be created so that each class contains all sequences of input combinations leading to a specific output value (or type). Other criteria could be based on coverage of specific states or transitions, paths within the state machine, etc.

The DFT approaches (as described in Section 2) help improve the performance of the testing procedure. In particular, the presence of memory elements in sequential circuits impairs the controllability and observability of internal signals, making testing more complex than in the case of combinational circuits. *Scan chain* [37] is a DFT technique that makes it easier to test sequential circuits when all the memory elements are connected to the chain (*full scan design*), allowing the use combinatorial tests instead of sequential pattern generation. Otherwise, sequential circuits without a full scan design require generating sequences of test vectors through the space of all possible vector sequences.

4 Game Model

We consider an *attacker* who inserts a single hardware Trojan in a digital IC. To minimize detection, the attacker does not insert multiple Trojans. The attacker's strategy consists of inserting a Trojan from one of the N possible classes defined in Section 3. Thus, the attacker has N strategies that we denote $S_1,\ S_2, \ldots,\ S_N$.

We consider that Trojans from different classes have different impacts in the system and thus different values for the attacker. The attacker's values for the Trojan classes are denoted by $V_1, V_2, \ldots, V_N (V_i \geq 0)$. We consider that the game is zero-sum. Any win for the attacker is a loss to the defender and vice-versa.

We consider a defender who has limited resources to test and detect hardware Trojans. In other words, the defender can only perform a partial test on each circuit. The defender tests the hardware for a limited number k of Trojan classes, $k < N$. Therefore, the defender has $\binom{N}{k}$ possible strategies. We assume that a Trojan is detected if the corresponding class is included in the subset tested; otherwise the Trojan is not detected.

Further, we consider that there is a fine $F\ (F > 0)$ that the attacker pays to the defender when a Trojan is detected. Unlike software attacks where attribution is difficult, in an IC attack, the hardware manufacturer is known and can be made responsible for Trojan infection by paying a fine. The new requirements from the Defense Logistics Agency (DLA) mandate that hardware manufacturers with DoD contracts must use a botanic DNA to mark their chips, which will increase the attribution reliability in hardware [38].

We consider the attacker and the defender to be both rational. The attacker inserts a Trojan that minimizes the likelihood of detection while the defender looks for a test set that maximizes the probability of detecting the Trojan. We assume that the test resources are common knowledge between the attacker and the defender. More precisely, both the attacker and the defender share the following knowledge:

- The defender's test size k.
- The attacker's values of the Trojan classes $V_1,\ V_2, \ldots,\ V_N$.
- The attacker's fine F.

In this model, the defender's tests cannot be deterministic because a rational attacker would simply avoid inserting a Trojan whose class is part of the test procedure. We propose to find the distribution that is most likely to detect a Trojan and yields the maximum payoff to the defender. Given this model, we have a strategic non-cooperative game having two players, i.e., an attacker and a defender. The attacker's strategy is to choose a Trojan to insert in an IC while the defender selects a subset of Trojans classes to be tested. Both players' payoffs will depend on the value of the Trojan inserted and the detection fines imposed on the attacker. We will investigate the mixed-strategy Nash equilibrium as a solution to this game. As previously mentioned, within the IC testing game, the use of mixed-strategies is suitable since the defender has an incentive to randomize over the test cases. Moreover, the mixed-strategy Nash solution will allow us to find the frequency with which the defender and attacker will choose certain strategies at the equilibrium. In a nutshell, in this robust testing game approach, a mixed strategy Nash equilibrium is still well-founded to fight irrational attackers or those with limited knowledge because any sub-optimum action they take yields a benefit to the defender, i.e., zero-sum.

5 Numerical Results

Without loss of generality, we consider a digital circuit with four input partitions $(N = 4)$. This leads to four classes of Trojans, thus the attacker has four different strategies. We denote the attacker's strategies by $A,\ B,\ C,\ D$. For this illustration, the values of the attacker's strategies are: $V_A = 1,\ V_B = 2, V_C = 4,\ V_D = 12$. However, the results in this section can be generalized for different choice of values V. The current values are chosen with different orders of magnitude of Trojans' impact on a system.

TABLE 1
Hardware Trojan Detection Game in Normal Form

		Defender					
		AB	*AC*	*AD*	*BC*	*BD*	*CD*
Attacker	*A*	$-F, F$	$-F, F$	$-F, F$	$1, -1$	$1, -1$	$1, -1$
	B	$-F, F$	$2, -2$	$2, -2$	$-F, F$	$-F, F$	$2, -2$
	C	$4, -4$	$-F, F$	$4, -4$	$-F, F$	$4, -4$	$-F, F$
	D	$12, -12$	$12, -12$	$-F, F$	$12, -12$	$-F, F$	$-F, F$

The defender tests 2 of the 4 possible Trojan classes, i.e., $k = 2$. Therefore, the defender has 6 possible strategies $(\binom{4}{2} = 6)$. The defender's strategies are: *AB, AC, AD, BC, BD,* and *CD*, as represented in the normal form game in Table 1.

Note that our model is still valid and will work when considering any other set of values for N, and k. The Nash equilibria are calculated using the game solver in [39]. Next, we will discuss the properties of the Nash equilibria and their implications on hardware testing. Section 5.1 elaborates on the mixed strategy Nash equilibria of the game in Table 1. Those mixed strategies will be used as the basis for testing hardware Trojans. Section 5.2 shows that the proposed approach is robust against irrational attackers. Section 5.3 considers an attacker who may choose not to insert any Trojan to avoid the risk of paying the fine after detection. Section 5.4 looks into a defender who would not test some of the circuits in order to save in time. Section 5.5 analyzes the attacker's payoff with respect to fine *F*. Section 5.6 scrutinizes the attacker's payoff depending on the number of Trojans tested by the defender. Section 5.7 analyzes the proposed game model in a situation where detection probability of each Trojan class is considered.

5.1 Mixed Strategy Nash Equilibrium

For this section and up until Section 5.4, we set the fine $F = 8$. Therefore, $V_A < V_B < V_C < F < V_D$. We will see in Section 5.5 (where the impact of changing fine *F* is analyzed) that it is not necessary for the fine to be greater than the biggest potential loss to deter a rational attacker from inserting Trojans in hardware. However, the minimum value of the fine should be set to yield a negative payoff for the attacker.

We can verify that the game in Table 1 does not admit a pure strategy Nash equilibrium. If the defender announces that he is testing only Trojan classes *C* and *D* (playing pure strategy *CD* may be because *C* and *D* are the two most dangerous Trojan classes), an intelligent attacker will insert a Trojan from class *B* which will go undetected giving the best payoff, i.e., a payoff of 2 for the attacker (thus -2 for the defender). The defender's best response to an attacker inserting a Trojan of class *B* is to play AB, BC or BD so that the Trojan can be detected and the attacker pays fine *F*, i.e., a payoff of -8 for the attacker and 8 for the defender. If the defender plays *BC*, the attacker's best response is to play *D* and go undetected, i.e., a payoff of 12 for the attacker and -12 for the defender. This circular reasoning shows that the game in Table 1 does not admit a pure strategy Nash equilibrium.

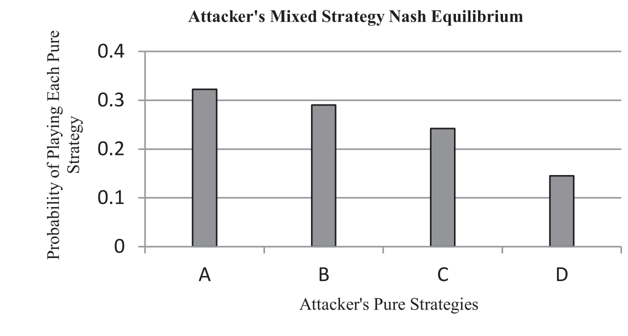


Fig. 3. Attacker's mixed strategy Nash equilibrium.

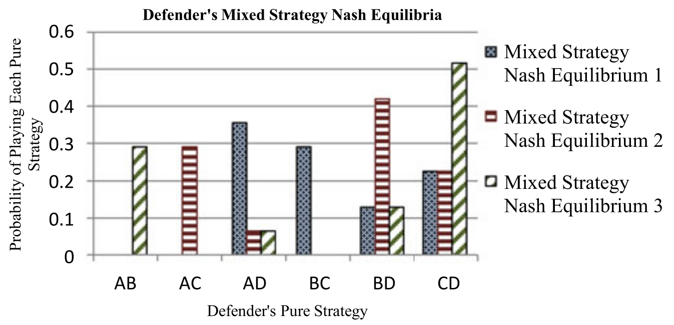


Fig. 4. Defender's mixed strategy Nash equilibria.

However, the game admits three mixed strategy Nash equilibria (as calculated by the game solver from [39]) that we denote (1), (2) and (3).

$$\{0.323A + 0.29B + 0.242C + 0.145D; 0.355AD + 0.29BC + 0.129BD + 0.226CD\}, \tag{1}$$

$$\{0.323A + 0.29B + 0.242C + 0.145D; 0.29AC + 0.065AD + 0.419BD + 0.226CD\}, \tag{2}$$

$$\{0.323A + 0.29B + 0.242C + 0.145D; 0.29AB + 0.065AD + 0.129BD + 0.516CD\}. \tag{3}$$

The attacker's strategy is the same for the three mixed strategy Nash equilibria. The mixed-strategy Nash solution for the attacker is shown in Fig. 3. On the other hand, the defender has three possible mixed strategy Nash equilibria represented in the clustered column of Fig. 4. In all the three Nash equilibria, the attacker's payoff is -2.193 while the defender's payoff is 2.193. The three Nash equilibria are payoff equivalent to each other. Both players are indifferent to which mixed strategy is used for testing hardware Trojans.

Fig. 3 shows a surprising result, i.e., the attacker's probability to insert a Trojan decreases with the value of the Trojan class. In other words, the attacker is less likely to insert a high value Trojan. This is because high values Trojans are heavily protected by the defender as shown in Figs. 4 and 5. A high value Trojan has more chances to get detected by a defender who adopts the strategic, game-theoretic testing procedure proposed here. Thus the attacker is better off inserting low value Trojans. This will result in the maximum possible benefit to the attacker who is taking into account the defender's best strategy.

Fig. 4 plots the three possible mixed strategy Nash equilibria of the defender. Mixed strategy 1 neither uses test *AB* nor *AC*. However, detection of Trojan classes *A*, *B* and *C* is covered by mixed strategy 1. Therefore, although combined classes AB and AC are not targeted by mixed strategy 1,

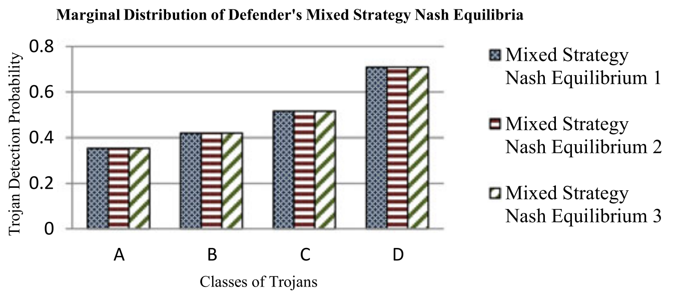


Fig. 5. Marginal distribution of the defender's mixed strategy Nash equilibria.

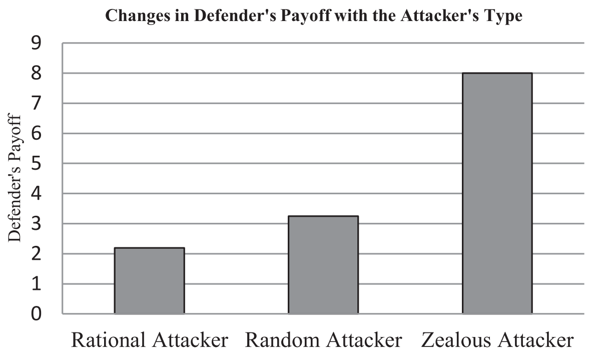


Fig. 6. Defender's payoff against rational and irrational attackers.

tests based on mixed strategy 1 can cover all individual Trojan classes A, B, C and D. We can make a similar conclusion for mixed strategies 2 and 3. Any of the three mixed strategy Nash equilibria cover all Trojan classes.

Fig. 5 expresses the marginal distribution of the defender's mixed strategy Nash equilibria. For instance, the defender's probability to detect a Trojan of class A is the marginal probability of A, which is calculated by adding the defender's probability of using a test containing A, i.e., AB, AC, and AD. Remarkably, we can see that the three mixed strategies have the same marginal distribution. Each test detects a given class of Trojan with the same probability. We will see in Section 5.3 that this result cannot be generalized. A Trojan with low impact, say of class A, is less often tested (35 percent of the times), while the high impact class-D Trojan is tested with a high probability, 71 percent. Finally, the sum of the marginal probability of A, B, C and D is 2 because the defender tests two Trojan classes at a time.

In short, although the defender has three mixed strategy Nash equilibria that appear to be completely different to each other, they share three important properties. First, they have the same marginal distribution. Second, the attacker's and defender's payoff are the same regardless of the mixed strategy used. Finally, each mixed strategy covers all classes of Trojans. Therefore, the defender can choose any of the three mixed strategies to optimally test for hardware Trojans.

Using the Nash equilibrium minimizes the risk of any possible exploitation by hardware Trojans. A defender who chooses one of the mixed strategy Nash equilibria, guarantees that he gets the maximum possible payoff against an intelligent attacker who, in turn, seeks to insert Trojans that escape detection and create the maximum damage. Therefore, the Nash equilibrium is a mathematically robust approach to detecting and defeating deceptive digital logic.

5.2 Case of Irrational Attackers

So far, we assumed that both attacker and defender act rationally. However, in practical cyber defense applications, one must consider how the solution of the game is impacted when the attacker acts irrationally. Indeed, irrational behavior in security and networked systems has become an area of high interest recently [22]. In this section, we will show that, for the studied model, irrational behavior will not have a detrimental impact.

We have shown in Section 5.1 that if both the attacker and defender are rational, and thus adopt a Nash equilibrium strategy, the defender's payoff is 2.193 while the attacker's payoff is -2.193. The defender has a strong incentive to use our test procedure (i.e., play one of the mixed strategy Nash equilibrium from Section 5.1). In fact, the attacker's and defender's payoffs remain the same if the defender uses one of our three tests, and the attacker chooses any pure strategy A, B, C, D or even plays a random uniform strategy or any other mixed strategy. This results from the definition of a mixed strategy Nash equilibrium and the fact that all four pure strategies A, B, C, D are part of the attacker's mixed strategy Nash equilibrium profile. With a mixed strategy, each player is randomizing in such a way that the opponent has the same payoff when playing any pure strategy. However, the defender can increase his payoff by playing his best response to an irrational attacker who does not play a Nash equilibrium strategy. Fig. 6 illustrates this result.

The first type of attacker we consider is a rational attacker who plays the mixed strategy Nash equilibrium shown in Section 5.1. The defender's best response will be one of the three mixed strategy Nash equilibria shown in Section 5.1 with the corresponding payoff of 2.193. Fig. 6 shows that only the rational attacker minimizes the defender's payoff.

The second type of attacker is a random attacker who plays the mixed strategy $0.25A + 0.25B + 0.25C + 0.25D$. The defender's best response is to play CD. Fig. 6 shows that this yields a higher payoff to the defender and thus a lower payoff for the attacker. A smart attacker will prefer to switch back to a mixed strategy Nash equilibrium.

Finally, we look into a zealous attacker who will always insert a class-D Trojan in an attempt to get the highest payoff. The defender's best response is to play one of the strategies AD, BD, or CD. Thus the attacker will be detected and pay the fine $F = 8$. This is the attacker's worst payoff.

5.3 Considering *No Trojan* as an Attacker's Additional Strategy

We saw that the game in Table 1 yields a negative payoff to the attacker, i.e., -2.193. Therefore, one may wonder why an intelligent attacker should insert any Trojan at all in the first place. We expand the game in Table 1 by adding a new strategy for the attacker, *No Trojan*. The defender's strategy and everything else remain the same. Moreover, if the attacker plays *No Trojan*, then both players get a zero payoff.

This new game has 42 Nash equilibria. Interestingly, the attacker plays the pure strategy *No Trojan* in all of those equilibria. However, the defender can choose among 42 different strategies, all of them mixed strategies. The defender successfully prevents an attacker from inserting a Trojan. Our testing procedure is then a form of cyber deterrence.

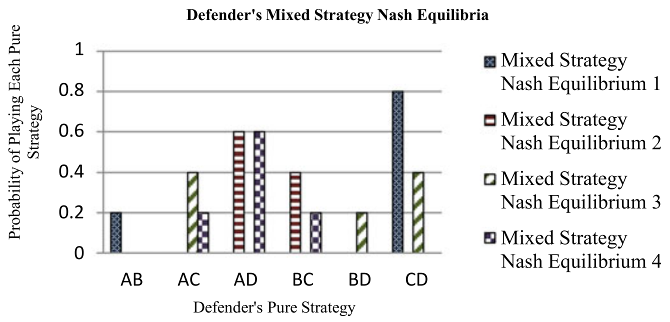


Fig. 7. Defender's mixed strategy Nash equilibria considering the *No Trojan* strategy.

For simplicity, we have selected only four of the 42 possible mixed strategy Nash equilibria available to the defender and represented them in Fig. 7. Note that the defender's mixed strategy Nash equilibria no longer have the same marginal distribution as opposed to the result in Section 5.1 (Figs. 4 and 5). Moreover, none of the three mixed strategy Nash equilibria in Fig. 4 remain an equilibrium for the new game. Therefore, a defender who considers the *No Trojan* strategy as a plausible attacker's strategy, must completely change his test procedure.

However, the set of 42 possible tests cover all Trojan classes *A*, *B*, *C*, and *D*. Finally, both players get a payoff of zero in all those 42 equilibria. This is because the attacker always plays *No Trojan*. If the fine *F* was such that the attacker had a positive payoff in the game of Table 1, then the attacker could adopt a mixed strategy for this new game in which he would sometimes insert a Trojan in the hardware.

5.4 Considering *No Test* Among the Defender's Strategy

The defender's goal for performing testing is to deter intelligent attackers from inserting hardware Trojans by making the attacks unprofitable. Therefore, a defender may want to conserve testing resources by letting some of the hardware go untested. This is the defender's *No Test* strategy. The attacker's strategy *No Trojan* remains an option in this section.

This new game has 62 Nash equilibria. As in Section 5.3, the attacker plays the pure strategy *No Trojan* in all of those equilibria. Moreover, the defender can choose among 62 different strategies, all of them mixed strategies. Notably, all of the 42 mixed strategy Nash equilibria in Section 5.3 remain

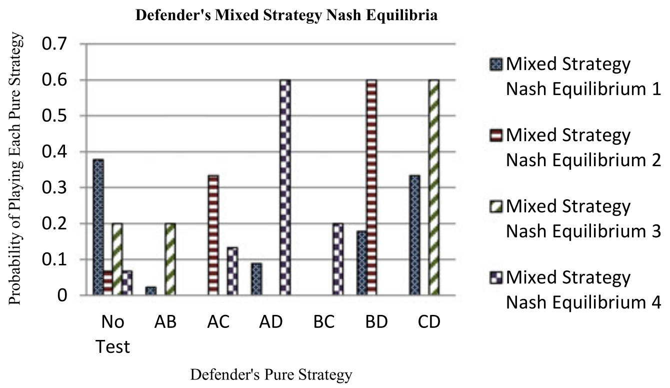


Fig. 8. Defender's mixed strategy Nash equilibria considering the *No Test* strategy.

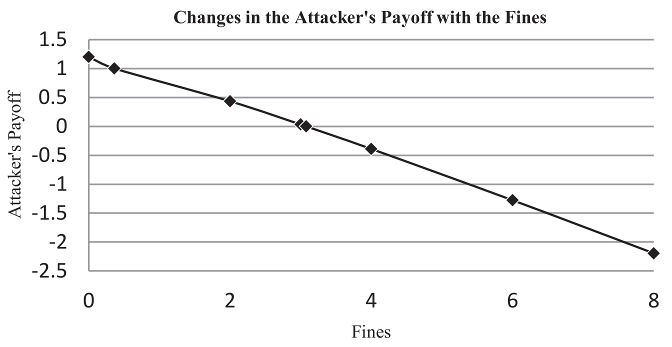


Fig. 9. Changes in the attacker's payoff with the fines.

as Nash equilibria for this new game. However, the defender has 20 new mixed strategies based on *No Test*. Fig. 8 shows 4 of the 20 new strategies. Both players still get a payoff of zero in all of those 62 equilibria. All Trojan classes are covered by the 62 equilibria.

Although the 62 tests have the same performance, the mixed strategy 1 of Fig. 8 will lead to the fastest test because it has the highest proportion of *No Test*. Using mixed strategy 1 as the test baseline, 37 percent of the hardware does not need to be tested. This results in a test procedure that is 37 percent faster than the 42 tests in Section 5.3.

5.5 Varying the Attacker's Payoff With the Fine *F*

Here, we consider the basic game of Table 1, i.e., without the *No Trojan* and *No Test* strategies and we vary the value of the fine *F*. We look into the changes of the attacker's payoff when fine *F* varies as shown in Fig. 9. We can see that the attacker's payoff decreases with *F*. The attacker has an incentive to insert a Trojan when fine $F < 3.078$ (positive attacker's payoff). For $F > 3.078$ (negative attacker's payoff), the attacker is better off not inserting any Trojan because the detection penalty might be too great.

Fig. 9 also shows that, when $F > 0.3643$, the attacker's payoff is less than 1. In contrast, if $F \leq 0.3643$, then the attacker's payoff is greater than 1 which means that inserting the Trojan *A* (recall that *A* has a value of 1) cannot be an optimum attack strategy. Thus *A* is not part of the support of the attacker's mixed strategy Nash equilibrium. The attacker's mixed strategy Nash equilibrium is to randomly insert one of the Trojans *B*, *C* or *D* according to the probability distribution shown in Fig. 10. This explains why Trojan class *A* is not used at all here, as opposed to Fig. 3 where the attacker uses *A* the most.

Since the attacker does not insert the Trojan *A* when $F \leq 0.3643$, the defender's best response is not to test *A*

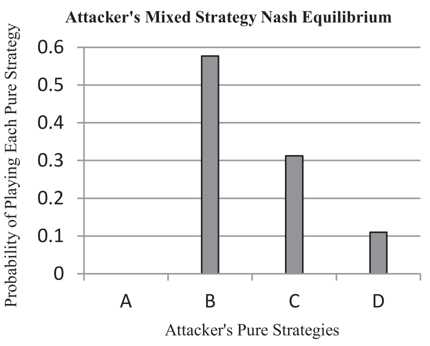


Fig. 10. Attacker's mixed strategy Nash equilibrium for $F = 0.3643$.

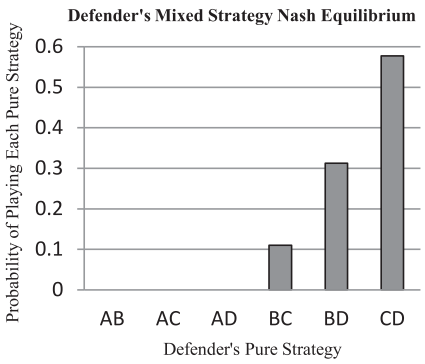


Fig. 11. Defender's mixed strategy Nash equilibrium for $F = 0.3643$.

because it will be a waste of testing resources. The defender's strategies *AB*, *AC*, *AD* cannot be optimum and are not part of the support of his mixed strategy. Instead, the defender's mixed strategy Nash equilibrium is to randomly use the test *BC*, *BD* or *CD* according to the probability distribution shown in Fig. 11. Surprisingly, the defender plays *BC*, *BD*, and *CD* with the same probability that the attacker plays *D*, *C*, and *B* respectively. That is because in the set {*B*, *C*, *D*}, the elements *D*, *C*, and *B* represent the complement of *BC*, *BD*, and *CD* respectively.

5.6 Variation of the Attacker's Payoff as the Number of Tested Trojans Changes

We consider fine $F = 8$ in this section and the basic game of Table 1. We consider defenders with different testing resources. A defender with no resources does not test for any Trojan while the attacker always inserts a class-*D* Trojan and gets a payoff of 12. A defender with full resour-ces always tests and detects all 4 classes of Trojans which yields a payoff of -8 for the attacker. Recall that testing two out of four classes of Trojans is the scenario we have analyzed above. It shows that the defender should test for at least 2 classes of Trojans to force an attacker to get a negative payoff and thus refrain from inserting hardware Trojans.

5.7 Consideration of Trojan Detection Probability of Each Trojan Class

So far, we have assumed that once a Trojan class is selected for testing, any Trojan existing in this class will be detected. Now, we introduce Trojan detection probability in our proposed game model and analyze its performance. Let's use P_A, P_B, P_C, P_D to represent the detection probabilities for Trojan class *A*, *B*, *C* and *D*, respectively. It is still a zero-sum game. Then, the game model in Table 1 will be modified as

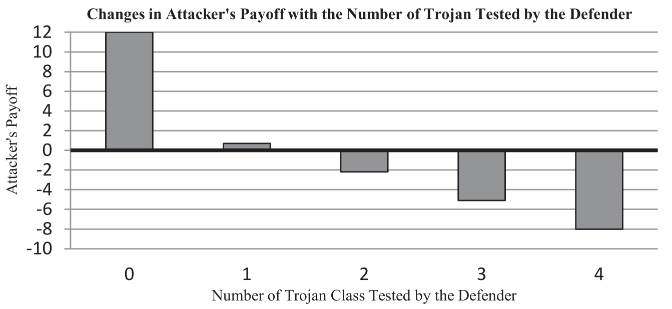


Fig. 12. Changes in attacker's payoff with the number of Trojans tested by the defender.

Table 2 with consideration of these Trojan detection probabilities:

Same reason as we explained in Section 5.1, Table 2 does not admit a pure strategy Nash equilibrium. But the game admits mixed strategy Nash equilibria. Let us consider detection probabilities for Trojan class *A*, *B*, *C* and *D* as $P_A = 0.6, P_B = 0.7, P_C = 0.8, P_D = 0.9$, respectively. To be consistent with our previous analysis, we still set $F = 8$, and the values of the attacker's strategies are: $V_A = 1$, $V_B = 2$, $V_C = 4$, $V_D = 12$. By using the game solver in [39], the game still admits three mixed strategy Nash equilibria as denoted as (4), (5), (6) shown below:

Three mixed strategy Nash equilibria considering Trojan detection probability

$$\{0.379A + 0.293B + 0.213C + 0.114D; 0.354AD + 0.282BC + 0.134BD + 0.229CD\}, \tag{4}$$

$$\{0.379A + 0.293B + 0.213C + 0.114D; 0.282AC + 0.072AD + 0.416BD + 0.229CD\}, \tag{5}$$

$$\{0.379A + 0.293B + 0.213C + 0.114D; 0.282AB + 0.072AD + 0.134BD + 0.512CD\}. \tag{6}$$

In all the three Nash equilibria, the attacker's payoff is -0.914 while the defender's payoff is 0.914. Similar analysis in the previous Sections 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6 could also be applied to this case which considers the probability of Trojan being caught when the corresponding test is included. Fig. 13 shows the same result as we see in the Fig. 3, that the attacker's probability to insert a Trojan decreases with the value of the Trojan class. Fig. 14 shows the three possible mixed strategy Nash equilibria of the defender, considering the Trojan detection probability. We could receive the same conclusion in Fig. 14 that any of the three mixed strategy Nash equilibria cover all Trojan classes.

TABLE 2
Hardware Trojan Detection Game in Normal Considering Trojan Detection Probability

		Defender					
		AB	*AC*	*AD*	*BC*	*BD*	*CD*
Attacker	*A*	$-P_AF+(1-P_A)V_A$, $P_AF-(1-P_A)V_A$	$-P_AF+(1-P_A)V_A$, $P_AF-(1-P_A)V_A$	$-P_AF+(1-P_A)V_A$, $P_AF-(1-P_A)V_A$	1, -1	1, -1	1, -1
	B	$-P_BF+(1-P_B)V_B$, $P_BF-(1-P_B)V_B$	2, -2	2, -2	$-P_BF+(1-P_B)V_B$, $P_BF-(1-P_B)V_B$	$-P_BF+(1-P_B)V_B$, $P_BF-(1-P_B)V_B$	2, -2
	C	4, -4	$-P_CF+(1-P_C)V_C$, $P_CF-(1-P_C)V_C$	4, -4	$-P_CF+(1-P_C)V_C$, $P_CF-(1-P_C)V_C$	4, -4	$-P_CF+(1-P_C)V_C$, $P_CF-(1-P_C)V_C$
	D	12, -12	12, -12	$-P_DF+(1-P_D)V_D$, $P_DF-(1-P_D)V_D$	12, -12	$-P_DF+(1-P_D)V_D$, $P_DF-(1-P_D)V_D$	$-P_DF+(1-P_D)V_D$, $P_DF-(1-P_D)V_D$

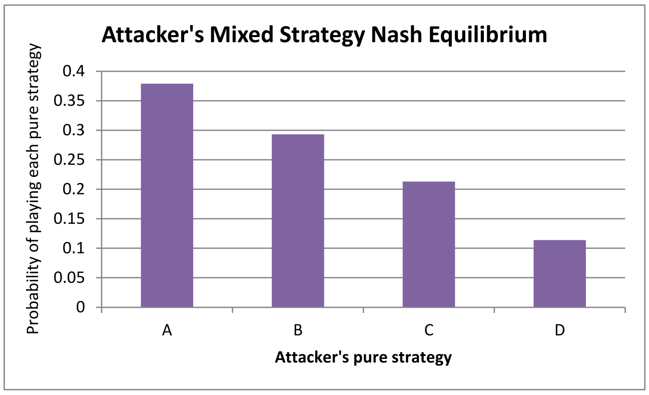


Fig. 13. Attacker’s mixed strategy Nash equilibrium considering Trojan detection probability.

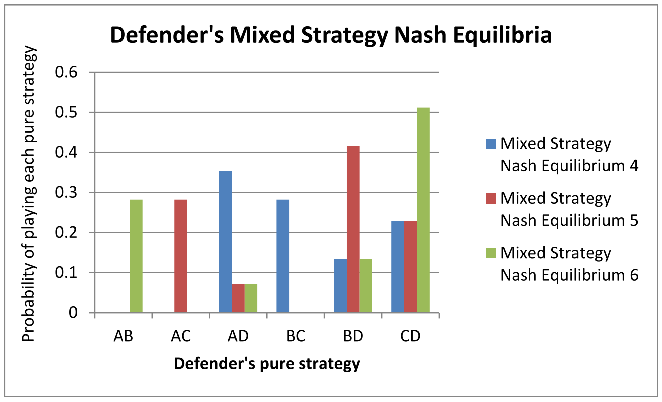


Fig. 14. Defender’s mixed strategy Nash equilibria considering Trojan detection probability.

6 Real World Scenario

The Trojan threat becomes more pronounced as ICs become more complex such as SoC (System on Chip). We consider, for a real world example, a SoC IC, and apply the proposed game-theoretic approach in our experiment. Our sample SoC IC is representing a typical SoC ICs in that it is built from third party IP cores and combined on a single chip. These IP cores may include embedded microprocessors, memory cores, interface cores, analog cores, networking cores and user defined cores that handle application specific processing functions [40]. The attacker can insert Trojans in any of these IP cores and each Trojan will have a different impact on the system. This raises serious concerns of trustworthiness of third party IP cores, and the defender is compelled to test the IC for the presence of Trojans. However, the defender is assumed to have limited resources to conduct all tests, so the proposed game-theoretic approach is applied to help the defender to identify the optimum test sets that will increase the probability of detecting and defeating Trojans while not exceeding the defender’s limited resources.

The architecture of the under-test SoC IC chip is shown in Fig. 15, which is hypothetically used in a critical missile-secure application. The Viper microprocessor core is included because it is primarily designed for defense and UAV applications. To provide security for the application, an application specific DSP core AES is included. The 10 Gigabit Ethernet Media Access Controller (10GEMAC) core is for supporting high-bandwidth demands of network traffic on LAN, MAN and WAN networks. The RS232 interface is implemented by a UART core. An analog IP core is to process mixed analog and digital signals. Any remaining user-defined specific functions are implemented in the Customer Proprietary Function Cores. The test module is to provide access to test these deeply embedded IP cores, and perform system level testing. All these IP cores are interconnected through standard AMBA (Advanced Microcontroller Bus Architecture) buses AHB/APB (Advanced High-performance Bus/Advanced Peripheral Bus).

In our proposed game model, an attacker is considered to have the ability to insert a single hardware Trojan in our SoC IC from N possible classes of Trojans. The defender will test k out of N Trojan classes. To be consistent with Section 5, Trojan classes are defined based on the different orders of magnitude of a Trojan’s impact on the system. Here, we adopt the existing best taxonomy from Trust-HUB [41] for classifying a Trojan’s effects on the system. There are four categories: Degrade Performance, Denial of Service, Change Functionality, and Leak Information. Thus, the attacker has four different strategies, which we denote as A, B, C and D, respectively as stated in Section 5. In this experiment, four Trojan benchmarks from [41] are considered, which cover all the possible effects a Trojan may have. These four Trojan benchmarks are for Viper processor, AES core, EthernetMAC, and the UART core. Table 3 shows the Trojan classes each benchmark belongs to, and the corresponding attacker values used in our experiment. The following explains these Trojan benchmarks:

- AES-T1000: The Trojan was inserted in the design and will cause the system to leak information. Whenever a predefined input plaintext is observed, the embedded Trojan leaks the secret key from a cryptographic chip running the AES algorithm through a covert channel.
- B19-T500: B19 is two copies of B14 (Viper processor) and two of B17 (three copies of B15—80386 Microprocessor) benchmark. In B19-T500, the Trojan trigger is a specific-vector (10110101) counter which resets with another specific vector (11111111). Whenever the counter value is between 100 and 110, the Trojan trigger becomes active and it manipulates the instruction register of the embedded Viper processor. The Trojan was inserted in the design and will cause the system to change functionality.
- RS232-T900: The Trojan trigger is a state machine inserted in the transmitter part of micro-UART core. The trigger seeks the sequence of 4 messages *8hAA*, *8h55*, *8h22* and *8hFF*. After Trojan activation, the Trojan payload blocks any transmission afterwards. The Trojan was inserted in the design and will cause the denial of service.
- EthernetMAC-T100: On the critical path, the net named “fault_sm0/n80” is widened. The attachment was compressed and divided into smaller parts to upload. It was inserted during fabrication and will cause degrade performance to the system.

When the attacker inserts a single Trojan (as described in the above Trojan benchmarks), the defender tests (due to the limited resources), only two out of these four Trojan classes at

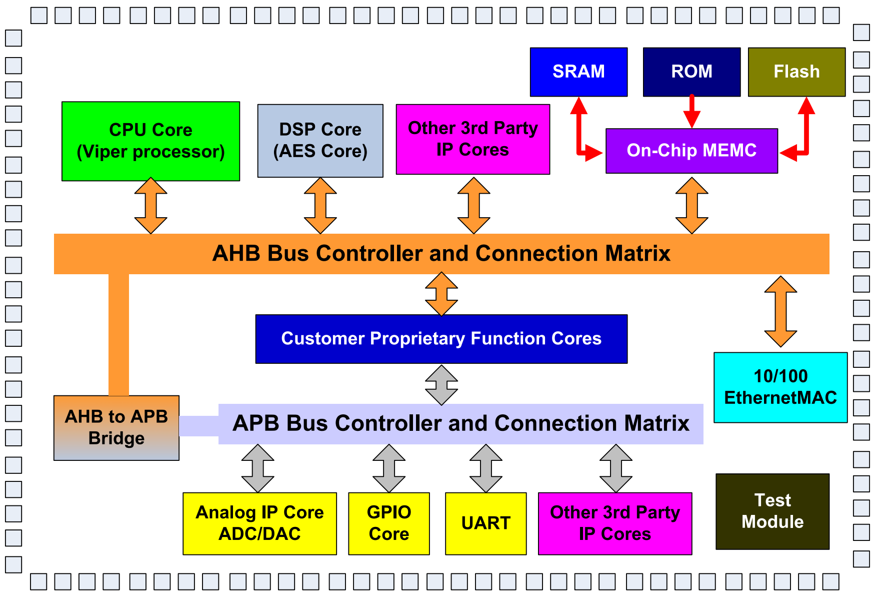


Fig. 15. The SoC IC architecture.

a time. If the Trojan is detected, then the attacker pays the fine F; otherwise the attacker receives a corresponding payoff. As stated in Section 4, we assume the Trojan is detected if the corresponding Trojan class is included in the two Trojan classes that are tested by the defender. The game model results in three mixed strategy Nash equilibria that allow the defender to identify optimally two Trojan classes to be tested in this experiment, which increases the probability of defeating the attack. The numerical results obtained in Section 5 can be applied in this SoC example because the same set of values for N, V_i, and k is used in the experiment. The attacker is less likely to insert a Trojan from category Leak Information, for instance, an AES-T1000, as it is considered the highest priority in a safety-critical application; it will be heavily protected by the defender who adopts the strategic game-theoretic testing procedure proposed in this paper. Thus, the attacker is better off inserting Trojans with lower values such as Degrade Performance, e.g., EthernetMAC-T100, to maximize the possible benefit to the attacker who takes into account the defender's best strategy. The effect caused by Trojan-EthernetMAC-T100 is considered less important in this SoC. For each of the three mixed strategy Nash equilibria of the defender, it covers all Trojan classes, though not all six possible strategies need to be addressed. This provides us optimum test sets that take into account the attacker's decision-making, and that will help increase Trojan detection coverage.

TABLE 3
Trojan Classes, Effects and Attacker's Strategy Values Used in the Experiment

Trojan Classes/ Attacker's Strategies	Effects	Trojan Benchmarks	Attacker's Values
A	Degrade Performance	EthernetMAC-T100	$V_A = 1$
B	Denial of Service	RS232-T900	$V_B = 2$
C	Change Functionality	B19-T500	$V_C = 4$
D	Leak Information	AES-T1000	$V_D = 12$

Please note that the number of Trojan benchmarks does not have to be limited to 4. For example, an attacker could insert a Trojan in any one of the IP cores, the AMBA buses in the SoC IC. We classify Trojans based on their effects to the system, which have four classes. Any Trojan that is inserted will fall into one of the four classes.

7 Conclusion and Future Work

We have proposed a game-theoretic methodology for analyzing the decision making processes involved in testing digital logic that may be infected with a hardware Trojan. The methodology allows for identifying optimum sets of test cases that take into account the attacker's decision-making, and that helps increase Trojan detection coverage. This is done via the calculation of the mixed strategy Nash equilibria that allow the tester of an IC to optimally combat Trojan attacks deployed by an intelligent attacker. The studied model also included the cases of: irrational attackers; attackers who may decide not to insert a Trojan in the circuit to avoid the risk of being caught and paying a penalty; and testers who may decide not to test a subset of chips to conserve testing resources. The proposed game-theoretic approach leads to three important properties that improve testing: (i) it allows to save testing resources by identifying test sets with the minimum number of test cases; (ii) when manufacturing large volumes of the same IC design, it allows to find the minimal subset of produced chips to be tested; (iii) it provides means to find the conditions that deter the attacker from infecting circuits with Trojans.

The methodology also analyzed changes in the attackers' payoffs when fines are levied for producing Trojan-bearing microcircuits, and when the tester increases (or decreases) the number of tests. An important result of our methodology is that it helps the defender find the minimum value for both the fine and the number of tests so that the attacker's payoff becomes negative and thus it discourages the attacker from inserting any Trojan at all, i.e., the *No Trojan* strategy (which is a pure strategy Nash equilibrium) becomes the best one for the attacker.

Future work will look into extensions for both the game theory model and the test procedure. For example, the game theoretic model can be extended to consider incomplete information games or lack of common knowledge between the attacker and the tester. We will also analyze the case in which a circuit infected with a Trojan does not provide reliable information about its manufacturer/designer (e.g., it has no botanic DNA) and thus the attacker cannot easily be identified. On the other hand, the test procedure can be extended with means for considering false positives/negatives and for estimating performance. Computational complexity of Nash equilibrium algorithms will be taken into consideration in regard of the size and complexity of the IC under test.

ACKNOWLEDGMENTS

Approved for Public Release; Distribution Unlimited: 88ABW-2015-3380, Dated 02 July 2015. This research was performed while Dr. Manuel Rodriguez held a National Research Council (NRC) Research Associateship Award at the Air Force Research Laboratory (AFRL). This research was supported by the Air Force Office of Scientific Research (AFOSR).

REFERENCES

[1] M. Tehranipoor and F. Koushanfar, “A survey of hardware trojan taxonomy and detection,” *IEEE Des. Test Comput.*, vol. 27, no. 1, p. 10–25, Jan./Feb. 2010.

[2] J. Rajendran, E. Gavas, J. Jimenez, V. Padman, and R. Karri, "Towards a comprehensive and systematic classification of hardware trojans,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 1871–1874.

[3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, “Trustworthy hardware: Identifying and classifying hardware trojans,” *IEEE Comput.*, vol. 43, no. 10, pp. 39—46, Oct. 2010.

[4] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. Hsiao, J. Plusquellic, and M. Tehranipoor, “Protection against hardware trojan attacks: Towards a comprehensive solution,” *IEEE Des. Test*, vol. 30, no. 3, pp. 6–17, Jun. 2013.

[5] C. A. Kamhoua, M. Rodriguez, and K. A. Kwiat, “Testing for hardware trojans: A game theoretic approach,” *in Proc. 5th Int. Conf. Dec. Game Theory Security*, Nov. 2014, pp. 6–7.

[6] K.-W. Lai and D. P. Siewiorek, “Functional testing of digital systems,” in *Proc. 20th Des. Autom. Conf.*, 1983, pp. 207–213.

[7] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, “Towards trojan-free trusted ICs: Problem analysis and detection scheme,” in *Proc. Des., Autom. Test Eur.*, 2008, pp. 1362–1365.

[8] S. Jha and S. Jha, “Randomization based probabilistic approach to detect trojan circuits,” in *Proc. 11th IEEE High Assu. Syst. Eng. Symp.*, 2008, pp. 117–124.

[9] R. S. Chakraborty, F. G. Wolff, S. Paul, C. A. Papachristou, and S. Bhunia, “MERO: A statistical approach for hardware trojan detection,” in *Proc. 11th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2009, pp. 396–410 .

[10] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, “Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically,” in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 16–19.

[11] C. Sturton, M. Hicks, D. Wagner, and S. T. King, “Defeating UCI: Building stealthy and malicious hardware,” in *Proc. IEEE Symp. Security Privacy*, 2011, pp. 64–77.

[12] H. Fujiwara, Logic Testing, *Design for Testability*. Cambridge, MA, USA: MIT Press, 1985.

[13] R. S. Chakraborty and S. Bhunia, “Security against hardware trojan through a novel application of design obfuscation,” in *Proc. Int. Conf. Comput.-Aided Des.*, 2009, pp. 113–116.

[14] H. Salmani, M. Tehranipoor, and J. Plusquellic, “New design strategy for improving hardware trojan detection and reducing trojan activation time,” in *Proc. IEEE Int. Workshop Hardware-Oriented Secur. Trust*, 2009, pp. 66–73.

[15] R. S. Chakraborty, S. Paul, and S. Bhunia, “On-demand transparency for improving hardware trojan detectability,” in *Proc. IEEE Int. Workshop Hardware-Oriented Security Trust*, 2008, pp. 48–50.

[16] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, “Identification and authentication of integrated circuits,” *Concurr Comput.: Practice Experiment*, vol. 16, no. 11, pp. 1077–1098, 2004 .

[17] K. Nguyen, T. Alpcan and T. Basar, “Stochastic games for security in networks with interdependent nodes,” in *Proc. Int. Conf. Game Theory Netw.*, 2009, pp. 697–703.

[18] C. Kamhoua, K. Kwiat, and J. Park, “A game theoretic approach for modeling optimal data sharing on online social networks,” in *Proc. 9th IEEE Int. Conf. Electr. Eng., Comput.Sci. Autom. Control*, 2012, pp. 1–6.

[19] S. Shiva, S. Roy, H. Bedi, D. Dasgupta, and Q. Wu, “A stochastic game with imperfect information for cyber security,” in *Proc. 5th In. Conf. i-Warfare Secur.*, 2010, pp. 308–318.

[20] Q. Zhu and T. Başar, “A dynamic game-theoretic approach to resilient control system design for cascading failures,” in *Proc. 1st Int. Conf. High Confidence Netw. Syst.*, 2012, pp. 41–46.

[21] W. Sun, X. Kong, D. He, and X. You, “Information security problem research based on game theory,” in *Proc. Intl. Symp. Public Electron. Commerce Security*, 2008, pp. 554–557.

[22] C. Kamhoua, N. Pissinou, and K. Makki, “Game theoretic modeling and evolution of trust in autonomous multi-hop networks: Application to network security and privacy,” in *Proc. IEEE Int. Conf. Commun.*, 2011, pp. 1–6.

[23] J. Jormakka and J. Molsa, “Modelling information warfare as a game,” *J. Inf. Warfare*, vol. 42, no. 2, pp. 12–25, 2005.

[24] Y. Liu, C. Comaniciu, and H. Man, “A bayesian game approach for intrusion detection in wireless Ad Hoc networks,” in *Proc. Workshop Game Theory Commun. Netw.*, 2006, Art. no. 4.

[25] C. Kamhoua, K. Kwiat, and J. Park, “Surviving in cyberspace: A game theoretic approach,” *J. Commun.*, vol. 7, no. 6, pp. 436–450, 2012.

[26] A. Agah, S. Das, K. Basu, and M. Asadi, “Intrusion detection in sensor networks: A non-cooperative game approach,” in *Proc. 3rd IEEE Int. Symp. Netw. Comput. Appl.*, 2004, pp. 343–346.

[27] J. Park, S. Kim, C. Kamhoua, and K. Kwiat, “Optimal state management of data sharing in online social network (OSN) services,” in *Proc. 11th IEEE Int. Conf. Trust, Secur. Priv. Comput. Commun.*, 2012, pp. 648–655.

[28] J. White, J. Park, C. Kamhoua, and K. Kwiat, “Game theoretic attack analysis in online social network (OSN) services,” in *Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Mining*, 2013, pp. 1012–1019 .

[29] J. Park, K. Kwiat, C. Kamhoua, J. White, and S. Kim, “Trusted online social network (OSN) services with optimal data management,” *Comput. Security*, vol. 42, pp. 116–136, 2014.

[30] L. Wang, S. Ren, B. Korel, K. Kwiat, and E. Salerno, “Improving system reliability against rational attacks under given resources,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 44, no. 4, pp. 446–456, Apr. 2014.

[31] X. Jin, N. Pissinou, S. Pumpichet, C. Kamhoua, and K. Kwiat, “Modeling cooperative, selfish and malicious behaviors for trajectory privacy preservation using bayesian game theory,” in *Proc. IEEE 38th Conf. Local Comput. Netw.*, 2013, pp. 835–842.

[32] C. Kamhoua, K. Kwiat, M. Chatterjee, J. Park, and P. Hurley, “Replication and diversity for survivability in cyberspace: A game theoretic approach,” in *Proc. Int. Conf. Inf. Warfare*, 2013, p. 116.

[33] C. Kamhoua, K. Kwiat, J. Park, P. Hurley, and M. Chatterjee, “Survivability in cyberspace using diverse replicas: A game-theoretic approach,” *J. Inf. Warfare*, vol. 12, no. 2, p. 27, 2013.

[34] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, “A survey of game theory as applied to network security,” in *Proc. 43rd Hawaii Int. Conf. Syst Sci.*, 2010, pp. 1–10.

[35] T. Alpcan and T. Başar, *Network Security: A Decision and Game-Theoretic Approach*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[36] N. Kukreja, W. Halfond, and M. Tambe, “Randomizing regression tests using game theory,” in *Proc. IEEE/ACM 28th Int. Conf. Autom. Softw. Eng.*, 2013, pp. 616–621.

[37] S. Narayanan, R. Gupta, and M. Breuer, “Optimal configuring of multiple scan chains,” *IEEE Trans. Comput.*, vol. 42, no. 9, pp. 1121–1131, Sep. 1993.

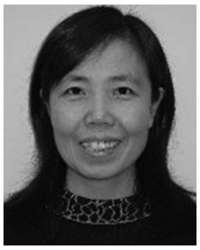
[38] C. Howard, “Counterfeit component chaos,” *Mil. Aerosp. Electron.*, vol. 12, p. 24 Dec. 2013.

[39] R. Savani. (2010). “Solve a bimatrix game,” [Online]. Available: http://banach.lse.ac.uk

[40] R. Saleh, et al., “System-on-Chip: Reuse and Integration”, in *Proc. IEEE*, vol. 94, no. 6, pp. 1050–1069, Jun. 2006.

[41] Available: https://www.trust-hub.org/taxonomy

Charles A. Kamhoua received the BS degree in electronic from the University of Douala (ENSET), Cameroon, in 1999, and the MS degree in telecommunication and networking and the PhD degree in electrical engineering from Florida International University, in 2008 and 2011, respectively. In 2011, he joined the Cyber Assurance Branch of the U.S. Air Force Research Laboratory (AFRL), Rome, New York, as a National Academies Postdoctoral Fellow and became a research electronics engineer in 2012. Prior to joining AFRL, he was an educator for more than 10 years. His current research interests include the application of game theory and mechanism design to cyber security and survivability. He has more than 50 technical publications in prestigious journals and International conferences including a Best Paper Award at the 2013 IEEE FOSINT-SI. He has been recognized for his scholarship and leadership with numerous prestigious awards including 30 Air Force Notable Achievement Awards, the 2016 Charles E. Perry Young Alumni Visionary Award, the 2015 AFOSR Windows on the World Visiting Research Fellowship at Oxford University, United Kingdom, an AFOSR Basic Research Award, the 2015 Black Engineer of the Year Award (BEYA), the 2015 NSBE Golden Torch Award—Pioneer of the Year, selection to the 2015 Heidelberg Laureate Forum, the 2011 NSF PIRE Award at the Fluminense Federal University, Brazil, and the 2008 FAEDS Teacher Award. He is currently an advisor for the National Research Council, a member of ACM, the FIU alumni association, and NSBE. He is a senior member of the IEEE.


Hong Zhao received the PhD degree from the New Jersey Institute of Technology in electrical engineering. She is an associate professor of electrical and computer engineering at Fairleigh Dickinson University, NJ. Her research interests include various aspects of broadband communications and computer security including network traffic/performance/security analysis and modeling, and hardware Trojan detection. She serves as associate editor of the *Journal on Multidimensional Systems and Signal Processing*, and editor of the *Journal of Computing and Information Technology*. She also serves as the chair of the IEEE North Jersey Computer Society Chapter, board member of a nonprofit organization, WOCC Inc., which promotes wireless and optical communications research and technical exchange. She has been a TPC member, symposium co-chair, technical paper reviewer, and book reviewer for IEEE conferences, journal magazines, and book publishers. She received the AFRL VFRP Award in 2014 and 2015, Visiting Professor Award from Ministry of Science and Technology in Taiwan, and 2015 IEEE Region 1 Award for Outstanding Support for the Mission of the IEEE, MGA, REGION 1, and Section. She is a senior member of the IEEE.

Manuel Rodriguez received the MS degree from the Polytechnic University of Valencia (UPV), Spain, and the PhD degree from the National Polytechnic Institute of Toulouse (INPT), France, im 1998 and 2002, respectively. He is a senior researcher at the Air Force Research Laboratory. From 1997 to 2002, he was a member of the Dependable Computing and Fault Tolerance group in the research laboratory LAAS-CNRS (Toulouse, France). He has participated in projects sponsored by government (AFOSR, NASA, and ESA), and industry. His research interests include security, dependability & fault tolerance, testing, formal methods, real-time systems, and software & hardware reliability.

Kevin A. Kwiat received the BS degree in computer science and the BA degree in mathematics from the Utica College of Syracuse University, and the MS and PhD degrees both in computer engineering and from Syracuse University. He is currently a principal computer engineer with the U.S. Air Force Research Laboratory (AFRL) in Rome, New York, where he has worked for over 33 years. He is currently assigned to the Cyber Assurance Branch. He holds four patents. In addition to his duties with the Air Force, he is an adjunct professor of computer science at the State University of New York, Utica/Rome, an adjunct instructor of computer engineering at Syracuse University, and a research associate professor with the University at Buffalo. He is an advisor for the National Research Council. He has been recognized by the AFRL Information Directorate with awards for best paper, excellence in technology teaming, and for outstanding individual basic research. His main research interest include dependable computer design.