University of Nevada, Reno

Think Smart, Play Dumb: A Game Theoretic Approach to Study Deception in Hardware Trojan Testing

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

by

Tapadhir Das

Dr. Shamik Sengupta - Thesis Advisor
May 2020

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

TAPADHIR DAS

Entitled

Think Smart, Play Dumb:
A Game Theoretic Approach to Study Deception in Hardware Trojan Testing

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Shamik Sengupta, Ph.D. , Advisor

Haoting Shen, Ph.D. , Committee Member

Sankar Mukhopadhyay, Ph.D. , Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

May, 2020

Abstract

In recent years, integrated circuits (ICs) have become a significant part in the operations for various industries and have given hardware security a greater priority, specifically in the supply chain where malicious manufacturers could insert hardware trojans (HT) to corrupt them. Due to budget constraints, many IC designers send ICs to offshore factories for manufacturing. When the designer gets the manufactured ICs back, it is imperative that they test for potential threats. In this thesis, a novel multi-level game-theoretic framework is introduced to analyze the interactions between a hardware manufacturer, who may be an attacker, and an IC designer, acting as defender, in terms of how they navigate the area of hardware testing. In particular, the game is formulated as a non-cooperative, zero-sum, repeated game using the mathematical framework of prospect theory (PT), which allows capturing the players' different rationalities when faced by uncertainty. The repeated game is separated into a learning stage, in which the defender learns about the attacker's strategy and an actual game stage, in which it acts accordingly. The thesis shows that there is a great incentive for the attacker to deceive the defender about their actual rationality by "playing dumb" in the learning stage. This scenario is captured by extending the game into a higher level in which hypergame theory is used to model the attacker's view of the game. To this end, the optimal deception rationality of the attacker is analytically derived to maximize the attacker's outcome from the deception process. For the defender, a first-step deception mitigation process is proposed to thwart the effects of deception. Simulation results show that the attacker can profit from the deception as it can successfully insert HTs in the manufactured ICs without being detected.

Dedication

This thesis project is wholeheartedly dedicated to my family: My dad, who has taught me that life is cold and unfair, and whatever happens, we must persevere and stand back up when we get knocked down. My mom, who has sacrificed immensely for the family, and frequently calls me to make sure I am doing well. My sister, who is one of the most psychologically resilient individuals I have ever known, inspires me every day.

I also would like to dedicate this project to all the wonderful friends I have made in graduate school. Whether it is for moral support, or we just need someone to talk to at 4 AM, having quality peers made my time here much more enjoyable.

Lastly, I would like to dedicate this thesis project to myself. If someone asked me five and a half years ago, when I was a freshman at Oregon Tech, "**Where do you see yourself in five years?**"; finishing my master's degree and starting my doctorate would have been the last thought on my mind. But, here I am. I am immensely fortunate and proud of the person I have become and am becoming.

Acknowledgments

Firstly, I would like to give immense thanks to my family: my dad, mom, and sister. They are my support system, and without their love, understanding, care, and sacrifices, I would not be in this situation.

I also am expressing my gratitude to my thesis advisor and soon-to-be doctoral advisor, Dr. Shamik Sengupta, who accepted me in his lab and gives me constructive and productive advice regarding my academic and professional life. I am also grateful to my TA supervisor, Dr. Candice Bauer, who is one of the most optimistic and knowledgeable individuals I know. Working as a Teaching Assistant under her has been a tremendous learning opportunity and has been extremely fulfilling. I would also like to thank Dr. AbdelRahman Eldosouky (Unfortunately, this doesn't count as a citation ☺) for his immense patience with me. I learned a lot from him, and I wish him the best for his future endeavors.

Lastly, no journey is worth the experience, unless you have good peers and friends who are embarking on it with you. I would like to thank all my friends and colleagues from the Computer Vision Lab, Trustworthy Systems Lab, and ENGR 301, as well as my friends Sanjeevan, Shuvo, Dusty, and Pourya. Lastly, a special thanks to my best friends: Michael, Jack, and Jonathan, who have been there for me since my high school and undergraduate days.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent years have seen a tremendous and unprecedented growth in technology. Innovations such as the Internet of Things, artificial intelligence, big data, and autonomous vehicles have taken over cyberspace. The evolution of these technologies has also led to the growth of electronic systems related to the ascent of these developments. One of the most impressive growths recorded in the past years is the manufacturing and usage of integrated circuits (ICs) in the field of technology. From the automotive and aerospace industry to the field of consumer electronics, transportation, and robotics, ICs are vital and an integral part of engineering systems [1]. Due to the rising cost of manufacturing, many prominent chip makers are outsourcing their designs elsewhere to help reduce the costs [2].

Outsourcing the manufacturing of ICs to third party vendors helps make manufacturing cost-effective for designers, but it also introduces serious security risks and threats [3]. Today, ICs constitute an integral part of manufacturing systems, consumer electronics, vehicular technology, communication systems, transportation systems, military technology, etc. As these are important aspects of a nation's economy, protection of ICs and hardware security have been given a larger priority than ever before. One serious threat faced in the field of hardware security and IC manufacturing is hardware trojans (HTs) [4]. A HT is a malicious design that can be added to the existing circuitry of an IC in order to corrupt its functionality.

HTs come with varying degrees of impact to an IC. Some HTs can cause error detection modules to accept inputs that should be rejected while some can downgrade the performance by intentionally corrupting a device's operational parameters. Certain trojans can leak sensitive data through both covert and overt channels or by creating a backdoor for malicious hackers into the IC [5]. Certain trojans can also generate a Denial-of-Service attack by targeting modules to exhaust scarce resources like bandwidth, computation, and battery power [6]. In short, hardware trojans are a serious security risk to an IC, and considering how important ICs are to most technological systems, it is of utmost importance to ensure their safety. The impacts of HTs are exacerbated even more when the infected ICs are used in devices connected to the Internet, which can facilitate cyber-physical attacks. Examples of these applications include hardware-dependent cognitive radios [7], IoT health systems [8], robotics [9], and time-sensitive unmanned aerial vehicles applications [10] and [11].

1.1 Related Works

HTs are designed to be stealthy, meaning that they cannot be easily found, and they might not be activated until a certain time, current, temperature, voltage, or logical factor is met [12]. Therefore, once the manufactured ICs are brought back to the designer, testing these circuits for potential security hazards is of foremost priority. There are multiple strategies that can be employed to test the prevalence of hardware trojans in an IC [4] [13]. In [4], the authors discuss two types of trojan detection techniques: destructive and non-destructive. Destructive techniques include gaining samples of the manufactured IC and completely de-metallizing them to compare before and after circuits, which is expensive and time consuming. On the other hand, non-destructive techniques involve side-channel analysis and logic testing. Due to the large area on an IC where trojans can be inserted, it is infeasible to generate all logical test benches to detect all types of potential trojans on the same IC. In [13], the authors used localized current analysis on certain portions of an IC to detect hardware trojans. As HTs require a power supply and a ground to function, any fluctuation from the normal operational current could be a sign for a potential HT. However, the methods in [4] and [13] rely heavily on the availability of adequate resources for testing and may be hindered by the lack of resources for effective testing. This raises the need for efficient testing strategies that can assist the tester to detect the most trojans under limited resources.

In order to combat the issues revolving around lack of testing resources, a promising approach that is recently being explored is studying the strategic interactions that may take place between an ICs manufacturer and a designer (tester). This can help the tester to efficiently use its resources to detect the most trojans. One effective method to carefully study the interactions between agents in a given situation is by using game theory. Game theory [14] provides a powerful mathematical tool to study such interactions and enables each party to achieve its best outcome in light of its opponents' actions. For instance, the works in [15] and [16] attempt to solve this hardware trojan testing scenario using game theory. In [15], the authors develop a “Trust Game" to illustrate the value of both the iterated elimination of dominated strategies (IEDS) and Nash equilibrium solution concepts. This work has been extended in [16] by computing multiple mixed strategy Nash equilibria which allows to effectively identify the optimal testing strategies to detect hardware trojans in a given IC.

The main assumption behind these game-theoretic frameworks is that the players involved are fully rational and are looking to maximize their expected utility against their opponent. In the real-world, players who face these situations rarely act rationally [17]. It was observed that players play irrationally when faced with obstacles or uncertainty, and they tend to deviate from their most rational choices. This phenomenon is best modeled using prospect theory (PT) [18]. The authors in [18] studied different applications of PT in the fields of finance, insurance, consumption-savings effect, industrial organization, labor supply, etc. They were able to show that users in all these domains were not fully rational under uncertainty.

PT can also be used with game theory in strategic decision making [19] and [20]. For instance, the authors in [19] applied PT to a zero-sum game between an attacker and a vendor of a drone delivery system in order to capture the subjective nature of the players with respect to attack success probabilities. In [20], PT was used to encapsulate the “user-centric" approach on microgrid power trading.

This use of PT was adopted in literature to study the HT problem under the case of limited rationality [12]. In particular, the authors in [12] formulated and analyzed a hardware trojan game, using a weighting effect of PT, in order to provide a subjective understanding of the interactions between the attacker and the defender. This application of prospect theory to the scenario of effective hardware trojan detection opened new doors into the research behind subjective human perceptions. However, one limitation of the work in [12] as well as the works [15] and [16] is that they do not take into consideration the concept of deceit. Deception, especially in security games, have attracted a lot of attention recently [21].

Deception refers to the act of intentionally behaving in a manner that is not consistent with one's true behavior. Deception is used in an effort to learn new information about other people or opponents. The idea of deception is based on the concept of misrepresentation. [22]. Misrepresentation of one's true intentions or capabilities are common in real world strategic interactions. The goal for this is to deceive one's opponent in order to have different perceptions about them than that are far from the truth. Multiple examples of this interaction can be found in [23] and [24]. For instance, the authors in [24], studied the impact of tactics and deception in chess. A player may deliberately sacrifice pieces at the start to see what their opponent's favorite piece is, or to see how calculative their opponent is. Then, they use this misrepresentation to their advantage during the endgame. Misrepresentation occurs in a multitude of other situations: wars, job applications, romantic courtships, etc. To this end, the problem of deception in HT games is a promising novel research direction that we are exploring in this thesis.

1.2 Contributions

The main contribution of this thesis is a general multi-level game-theoretic framework to study and model deception in hardware trojan detection systems. Unlike the prior work of [12], [15] and [16], our framework is used to model and analyze the possible deception actions of an attacker in HT testing games. The framework is built on different levels based on the players' rationalities and the deception strategy.

In the proposed framework, we use game theory to model a non-cooperative, zero-sum game between the attacker and the defender, based on the available strategies for each player. This game theory model represents the first level of the framework and is based on the works of [16] and [12]. In the proposed game, we assume that the attacker is aiming to maximize its expected utility by inflicting maximum possible damage using hardware trojans. The defender, on the other hand, is trying to limit this damage by developing effective testing strategies to thwart the attacker's trojans. If the attacker is successful, it receives a utility based on the type of trojan used. If the defender is successful, a fine is imposed on the attacker.

The next level of the game accounts for the players' rationalities. In particular, the players' strategic profiles are weighted according to the players' rationalities. Note that, unlike the work in [12], which proposes a HT game solution using prospect theory, our framework uses PT utilities and the weighting effect as a step towards studying the deception. In this level of the game, prospect theory is used to capture the players' strategic deviations and subjectivity under uncertainty and risk.

The whole game is, then, formulated as a repeated game where the static game, previously defined, is played over multiple stages. The repeated game resembles the idea of ICs returning from the manufacturer in different batches, and each batch contains multiple ICs that need to be tested. Due to the massive IC space and limited resources, it is impossible for the defender to test for every single potential trojan type on every IC in every batch. We propose that in such scenarios, a promising approach for the defender is to learn about the attacker's strategies in order to develop strategic testing patterns and apply this learning to subsequent stages of the game. Because of this, our whole game is broken down into two separate stages: the learning stage and the actual game stage. The learning stage is where the defender learns about the attacker's inclinations and preferences. This knowledge is then applied to the rest of the actual game stage.

However, due to having different stages in the repeated game, it becomes motivating for the attacker to try to deceive the defender. Here, we propose a new type of deception called *rationality deception attacks*. In this type of attack, the attacker will play with a lower rationality than its true rationality during the learning stage. In a sense, it will be “playing dumb" in order to deceive the defender during the learning stage. That makes the defender think that the attacker has a lower rationality, and thus, their actions are not aligned with its interests. The defender will then focus on that respective attacker strategy profile, which allows the attacker to play at a higher rationality (actual rationality) than what the defender is expecting during the actual game stage. To the best of our knowledge, this is the first work to consider deception through manipulating the rationality levels in PT.

Subsequently, to capture this misrepresentation and deception in game scenarios, an additional level of the game is presented. In particular, the framework of “hypergame theory" is used [22]. Hypergame theory is an expansion on traditional game theory. In these circumstances, there are additional levels of game play and utilities on top of a base level. The base level of the game can be any common game theory situation like the classifications discussed in Sections 2.2. However, the additional levels and their associated utilities are only aware to a subsection of the total number of players. This is a framework used to create an unbalanced game scenario, so not all players have the same view of the game being played. Hypergame theory is structured as a hierarchical game where certain players have an extended view of the game being played from their opponents. The opponents, in this case, may or may not have any idea about this extended view of the game and have a perceived partial view of the same game. In this thesis, hypergame theory is used to analytically derive the optimal deception rationality of the attacker, i.e., the lower rationality that the attacker will pretend it has.

Finally, we propose a first-step defense technique for the defender to mitigate the effects of the deception in case it is unaware of the occurrence of the deception. More effective techniques are referred to but are left for future work. We show through simulations that the attacker can benefit from the proposed deception attack as it can insert HTs into the manufactured ICs without being detected. Simulation results are also used to study the attacker's deception utilities under different combinations of the attacker's and defender's rationalities.

The rest of this thesis is organized as follows: Chapter 2 provides a basic understanding and overview about the concepts involved in this thesis including a brief introduction to game theory, types of games, nash equilibrium, and mixed and pure strategies. The system model is highlighted in Chapter 3. Chapter 4 demonstrates the first two levels of the game, which are the non-cooperative zero-sum static game and the extended game formulation using prospect theory utilities. In Chapter 5, the deception attack is introduced along with the third level of the game using hypergame theory. This Chapter concludes by introducing a basic defense mechanism to thwart a deceptive attacker. Numerical results and simulations are presented and analyzed in Chapter 6. Finally, conclusions are drawn in Chapter 7.

Chapter 2

Background on Game Theory

Game theory is a popular approach that is employed by many fields in order to study effective and strategic communications and interactions between agents in a scenario. This assist involved parties to understand the best outcomes or the best way to manipulate the strategies at hand to optimize their outcomes. Game theory is prominently used in computer science for situational study. In this chapter, we present a basic overview of game theory and its associated concepts.

2.1 Game Theory

Game theory is a robust algorithmic framework that is utilized to study situations and circumstances where agents are interacting with each other and every agent is attempting to optimize their utility in light of the other agent's actions. [25], [14], [26]. This field can be used to model situations where players are in conflict with one another or try to cooperate with one another to find a productive result [27]. Game theory can be and has been used in various disciplines expanding from political science and economics, to social science and psychology [28], [29], and [30]. In the field of computer science, game theory has shown to have a wide variety of applications including decision making in cybersecurity, networking, resource allocation, electronic commerce, artificial intelligence, multi-agent systems, information sharing, etc [31].

A game consists of players, their associated strategies, and the corresponding utilities to said strategies. The utility functions for each player is based on a mathematical function that represents the rewards received or costs incurred by the player based upon their played strategy vs the opponent's strategies. All players in a game scenario are assumed to be completely rational, meaning that they are always trying to optimize their utility [32].

2.2 Classification of Games

This section showcases common classification of game types.

2.2.1 Cooperative vs Non-cooperative

A prominent classification of game types in game theory is cooperative vs non-cooperative games [33]. In non-cooperative games, players play a game scenario against one another where each player is trying to optimize their utility considering their opponent's strategies. On the other hand, cooperative games involve players playing or “cooperating" with one another to optimize the collective utility of the players. Cooperative game theory studies player coalitions and resource sharing and tries to find fair outcomes of these games [34].

2.2.2 Complete vs Incomplete Information Game

Games can also be classified as complete vs incomplete information games. In complete information game scenarios, all players have complete information about every other players' strategies and utilities [35]. In incomplete information games, this framework is not followed, and players don't have all information regarding other players' strategies and utilities [36].

2.2.3 Pure vs Mixed strategy

Pure strategy is a strategic profile for a player if the player plays a certain strategy with a probability of 1.0, which means this particular strategy is the absolute strategy that will be played. On the other hand, a player plays a mixed strategy when they play a probability distribution over their available strategies.

2.2.4 Static vs Repeated Games

A static, or one-shot, game is a game format where players can only make decisions once. The players play simultaneously together, so that there is no “learning" that can be achieved from either player's perspective unlike a repeated game. A repeated, or dynamic, game is a game scenario where an original static game is repeated N times. The mixed strategies of players in a repeated game stage may be dependent on previous strategies that their opponents played.

2.3 Nash Equilibria

In standard game theory, Nash equilibria is defined as a joint-action strategy for all players in the game from which it is not possible for any player to unilaterally deviate and further optimize their utility. It is also referred to as the stable solution of the game. Nash equilibria are usually represented as the steady state of a game scenario and are most times, the predicted solution of the game [37].

Chapter 3

System Model

Consider an integrated circuit (IC) manufacturing company, referred to as the “attacker", that has an incentive to attack a designing company, which is labeled as the “defender" from here. We assume that the attacker can insert one trojan t from a set of $\mathcal{T}$ trojan types. Each trojan $t \in \mathcal{T}$ leads to a certain damage and in turn, provides a respective utility, V_t, to the attacker if the trojan was not detected by the defender. Due to the unique operational parameters of each trojan t like voltage, current, power, access to certain modules , etc., every trojan can only be inserted in a unique partition on the IC.

After getting the designed ICs back from the manufacturing company (the attacker), the defender's job is to test the ICs for potential threats like hardware trojans. As modern day ICs are extremely complex, it is demanding and very resource expensive to test for every kind of potential trojan on every single IC. Due to being limited by testing resources, the defender can only test for a certain number of trojans per IC, which is a common assumption in literature [12] and [16]. To this end, let $\mathcal{A}$ be the tester's subset of trojans that can be tested at a time, such that $\mathcal{A} \subset \mathcal{T}$. This subset

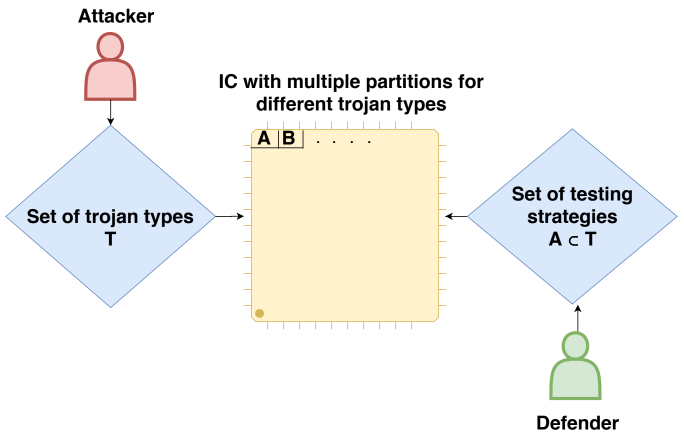
Attacker
IC with multiple partitions for different trojan types
A
B
. . . .
Set of trojan types T
Set of testing strategies A ⊂ T
Defender

Figure 3.1: Attacker-Defender Game Scenario

will include all different combinations of trojans that can be tested simultaneously, based on the tester's capacity.

If the defender successfully detects the presence of a hardware trojan (HT), the attacker incurs a fine of F_t where t refers to a trojan type and $t \in \mathcal{T}$. The magnitude of the fine could be represented as legal consequences for trying to infect an IC with a certain trojan type. Types of legal consequences and fines could range from paying a monetary amount for damages to the termination of the contract between designer and manufacturer [12]. In this model, we limit F_t to monetary fines as in a real-life scenario, it will take the designer a long time to terminate the contract and to shift the production to another manufacturer.

Fig. 3.1 shows an illustration of the game model in which each IC is partitioned into different sectors and trojans can be inserted into these sectors. Both the attacker and the defender have their own sets of strategies that correspond to attacking and defending against specific trojans, respectively, where each trojan is unique for a specific sector.

To understand the interactions between the attacker and the defender, we use game theory to study the interactivity. The goal is to mathematically model the interactions between the players to find methods or strategies that can be used to insert certain trojans inside an IC, and how these attack strategies can help the defender develop testing patterns and strategies to impact the overall damage to the system. Using this approach, optimal testing strategies for the defender can be solved to the various attacking strategies from the attacker.

Chapter 4

Game Formulation

In this game, all players are attempting to gain the upper hand against their opponents. For the attacker, it is trying to play its best strategy based on its perception of the kind of testing strategy the defender will try to employ. For the defender, it is trying to play its best testing strategy based on its perception of the attacker's tendencies and potential strategies that it may employ. The strategies that are employed by both players end up leading to either corrupting the integrated circuit (IC) or levying a fine for the attacker. Here, the game can be modeled as a non-cooperative game. We will also consider both a single shot game, i.e., a static game, and the case where this static game is repeated over a certain number of batches. This represents a scenario where manufacturer is supplying the chips back to the designer in multiple batches.

4.1 Static Game

We consider two players: the attacker a and the defender d in a set $\mathcal{N}_P$ such that $\mathcal{N}_P := \{a, d\}$. Let the set $\mathcal{S}$ represent the strategy space, $\mathcal{S}_d$ and $\mathcal{S}_a$, of the defender and the attacker, respectively. These strategy spaces represent all the possible actions for the players. Let the set $\mathcal{U}$ represent the utility functions of the players, U_d and U_a, for the defender and the attacker, respectively. Finally, let the game $\mathcal{G} = \{\mathcal{N}_P, \mathcal{S}, \mathcal{U}\}$.

For the attacker, the strategy space consists of all the different kind of trojans that can be played in this game, i.e, $S_a = \mathcal{T}$. Here, every strategy in the attacker's strategy space , $s_a \in S_a$, refers to a corresponding trojan type where $t \in \mathcal{T}$. We assume that the attacker is always inserting a trojan from the corresponding set of available trojans. Thus, we can capture the maximum damage an attacker can cause to the defender.

The defender, on the other hand, will select a subset of trojan types to test simultaneously per stage of the game due to the limited resources and the large number of trojan types that can be tested for. Let the number of trojans, that the defender can test at once, be K types of trojans. The value of K is proportional to the amount of resources available at the defender's disposal for trojan testing. The strategy space of the defender can then be $\mathcal{S}_d$, defined as all the possible subsets of $\mathcal{T}$ with the size of K, i.e, $\mathcal{S}_d = \binom{\mathcal{T}}{K}$. Here, every subset possibility is present in the defender's strategy space ,$s_d \in \mathcal{S}_d$.

Players' utilities can be determined using the strategy selection for the attacker, s_a, and the corresponding strategy selection of the defender s_d such that:

$$U_a(s_a, s_d) = \begin{cases} -F_{s_a} & \text{if } s_a \in s_d, \\ V_{s_a} & \text{otherwise,} \end{cases} \tag{4.1}$$

where V_{s_a} is the attacker's gain for playing a certain strategy s_a that was not detected by the defender strategy, s_d. $-F_{s_a}$ is the attacker's fine for playing a certain strategy s_a that was detected by the defender strategy s_d. The magnitude of V_{s_a} reflects the monetary reward gain by the attacker, which also corresponds to the type of damage that the trojan s_a can cause, if the attack is successful. On the other hand, the attacker's fine $-F_{s_a}$ reflects the penalty that the attacker incurs if s_a is detected. The magnitude of the fine could represent consequences ranging from paying a monetary amount for damages to the termination of the contract between designer and manufacturer [12]. However in this model, we limit F_{s_a} to monetary fines as in a real-life scenario, it will take the designer long time to terminate the contract and to shift the production to another manufacturer. We notice that the outcome of the game will be either a fine F_{s_a} charged from the attacker and paid to the defender, or the attacker's gain V_{s_a} , which represents the defender's loss. In this case, the utilities are exactly opposite to one another, meaning, that the exact reward which the attacker receives is the exact amount the defender loses, and vice-versa. Thus, the game will feature a zero-sum characteristic and the defender's utility can then be given as:

$$U_d(s_a, s_d) = -U_a(s_a, s_d). \tag{4.2}$$

Finally, let $P = \{p_a, p_d\}$ represent the mixed strategy for the attacker and the defender respectively, which represents the players' objective probabilities of selecting actions from their strategy spaces to maximize their expected utility.

In addition to the utilities defined in equations (4.1) and (4.2), we also focus on the concept of prospect theory (PT) [18]. Under normal expected utilities, players tend to play their most optimal strategy to increase their expected utility. However, according to PT, players deviate from their most rational strategies and maximized utility potentials when faced with uncertainty regarding strategies, or if there are limitations to playing a certain strategy that is not taken into account during the game. Under PT, players have a subjective rationality of the opponent's strategies, and hence, change the expected utilities from an objective to a subjective one.

In this particular game, both players are facing uncertainty when it comes to their opponent's strategies. For the attacker, they are not completely sure of the testing strategy that the defender will employ. Therefore, the attacker may tend to underweight or overweight a particular strategy of their opponent. As per the defender's point of view, they are not sure of the attacker's tendencies and actual preference in trojan types, so the defender may underweight or overweight certain attacker strategies as referred in [12]. Also, as both players are human, there could be multiple reasons for the subjective rationality of the opponents: company regulations, requirement of more resources to play a certain trojan, not enough resources to test for certain trojans, lack of knowledge on opponent's strategies, etc.

In order to capture the deviation from the optimal strategy for each player, a *weighting effect* w is incorporated. Under this weighting effect w, players give a subjective weight to their opponent's strategies for more relevance. Another incorporated parameter is the *rationality parameter* α, which judges a player's subjective perception based on their objective probability between $0 < \alpha \leq 1$ where a player's rationality of 1 means they are playing with full rationality, i.e., complete objectivity. This rationality level captures the nature of human subjectivity when faced with uncertainty. In this case, the Prelec function [38] can be used to capture the weighting effect for every player's observed strategies and is defined as:

$$w_i(p_i, \alpha_i) = \exp(-(-\ln p_i)^{\alpha_i}), 0 < \alpha_i \leq 1. \tag{4.3}$$

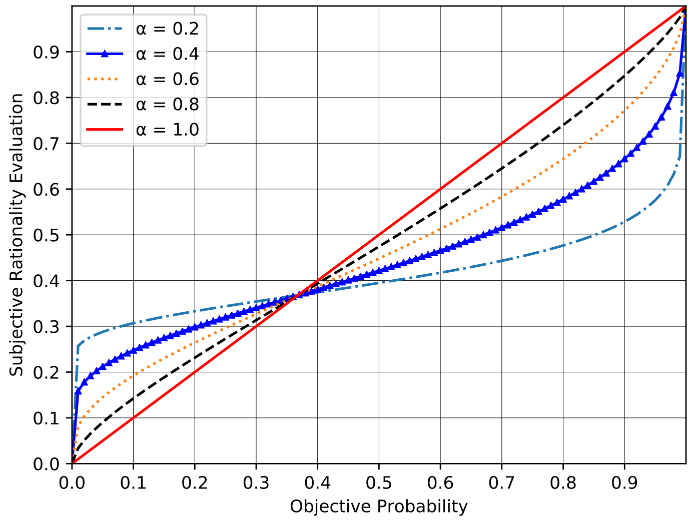

Figure 4.1: Player objective probability vs their subjective rationality evaluation of a strategy

The Prelec or Probability Weighting Function is used to highlight human subjectivity, where players assign a subjective weight to the likelihood or objective probability that an event may occur. This weight is based on player rationality α_i, which is integral in calculating the weighted probability distribution from an objective probability distribution. This turns an objective probability distribution into a subjective one, depending on the player's rationality α_i. Fig. 4.1 shows the impact of α_i on the distortion of the player and highlights how rational player i is by measuring the uncertainty and risk faced by the player regarding the opponent's probability.

Using the Prelec function, the utility for every player can then be updated with respect to their perceived rationality about their opponent's probabilities as follows:

$$U_i^{PT}(p_i, p_j, \alpha_i, \alpha_j) = \sum_{s=S} \Big(p_i(s_i) w_i \big(p_j(s_j)|_{\alpha_j}, \alpha_i \big) \Big) u_i(s_i, s_j), \tag{4.4}$$

where i and j correspond to both players. Equation (4.4) states that each player will use the Prelec function to weigh their opponent's probabilities, which in turn was calculated using their opponent's rationalities. The players gradually learn about their opponent's rationality during the learning stage of the game.

Based on the defined utilities in (4.4), each player will take actions by selecting a trojan or a subset of trojans to maximize its utility given the actions of the other player. However, as one player maximizes its utility, it will hurt the other player's utility, which will result in changing their actions to improve their utility. This will continue unless the players are able to find an equilibrium solution [14]. Equilibrium solutions, in game theory, are referred to as Nash equilibrium, which occurs when no player can improve its utility by unilaterally changing its actions, and hence, Nash equilibrium presents a stable solution for the game [39]. Nash equilibrium can either be pure Nash equilibrium when every player chooses only one action, or mixed-strategy Nash equilibrium, which is a probability distribution over the player's set of actions. Here, we focus on mixed-strategy Nash equilibrium as it presents a general solution for the equilibrium. Each finite game, like the game defined in this section, is known to have at least one Nash equilibrium solution. We study this equilibrium solution in Chapter 6. The sensitivity of this resultant Nash equilibrium solution to other variable parameters in the game scenario is not studied in this thesis. However, this analysis can be left for future work, and it can be done similar to the work in [12].

4.2 Repeated Game

Here, we consider the case where the manufacturer provides the ICs to the designer in different batches. Each batch consists of multiple identical ICs. Checking all the ICs in one batch, for all types of HTs, is a time consuming and expensive process for the defender. Therefore, a promising approach for the defender is to learn about the attacker's strategies in order to develop strategic testing patterns and, then, apply this learning to subsequent batches. This means that it would be beneficial for the defender to check every single IC for every possible HT, in the initial set of batches, to figure out the attacker's probabilistic preferences for choosing HTs. Then, the defender can use this knowledge to test for HTs in subsequent batches after learning about the attacker's preferences. This scenario can then be modeled as a repeated game [14]. Here, we propose that the game can be divided into two separate stages: the learning stage and the actual game stage.

During the learning stage, the defender will check every single IC in each batch. This will help the defender to update its belief about the attacker's strategies, which correspond to its rationality level. In the game stage, the defender will use this belief to save time and to check a few ICs, instead of all the ICs, per batch. The defender will focus on checking the types of trojans that correspond to the attacker's strategy that was learned from the learning stage. The number of batches of the learning stage will depend on the defender's choice based on the defender's confidence about the belief update.

Note that, under this scenario, each player will take actions twice. Once at the beginning of each game stage. These actions will affect the player's outcome from all the subsequent batches after taking these actions. Let N be the total number of batches in the game, which corresponds to the number of the batches that will be delivered by the manufacturer. For every batch in the learning stage, the defender is going to check all ICs per batch for hardware trojans, which is denoted by C_L. In the game stage, the defender will randomly select a few ICs per batch to test for hardware trojans, denoted by C_A such that $C_A < C_L$. In every IC, the defender will look for the same number of HTs, given by K. The defender will choose N_L to be the number of learning batches and N_A to be the number of batches in the game phase, where $N_A = N - N_L$. The utility in the learning stage , U_L, will be computed by:

$$U_{L_i} = C_L U_i^{PT}(p_i, p_j, \alpha_i, \alpha_j), \tag{4.5}$$

Similarly, the utility in the actual game stage will be given by:

$$U_{A_i} = C_A U_i^{PT}(p_i, p_j, \alpha_i, \alpha_j). \tag{4.6}$$

The total utility U_{T_i} of the entire game will be given by:

$$U_{T_i} = U_{L_i} + U_{A_i}. \tag{4.7}$$

where i denotes a player in this game: attacker or defender. The equilibrium of this repeated game is discussed in details in Chapter 5.

Chapter 5

Hypergame Model for Deception in Hardware Trojan Games

Many cybersecurity games follow the concept of repeated games. Many of these scenarios involve a learning stage and an actual game stage. Players assume that their opponent is letting out their truest intentions and are consistent with an observed strategy and, in turn, will continue to play this strategy in subsequent stages of the game. Cunning players, however, can try to exploit this assumption by deceiving their opponent between game stages.

Here, we notice from Fig. 4.1 that according to a player's rationality, it will weigh the probabilities in a different order between low and high probabilities. The inflection probability from Fig. 4.1 is about 0.37. For instance, a probability of 0.1 will be weighted higher for low rationality levels, e.g., 0.2 than for a rationality of 0.8. On the opposite side, a probability of 0.7 will be weighted higher for a rationality of 0.8 than for a rationality of 0.2. This gives an incentive for the attacker to misrepresent itself and play a different rationality during the learning stage to deceive the defender and play its actual rationality in the rest of the game. In this case, the attacker can try to deceive the defender by playing a lower rationality or , “acting dumb", during the learning stage, and then playing its actual rationality in the actual game stage. This misrepresentation of one’s true tendencies is prevalent in the world of interaction, and capturing it is the focus of this thesis. Here, we propose to use hypergame theory [40] as a prominent framework to model such deceptive interactions.

5.1 Hypergame Theory

A prominent section of game theoretic scenarios deals with complete information. This means that all players are aware of all strategies, for themselves and their opponents, and the respective utilities associated with said strategies. However, in the case of deception, we assume an additional level of utilities. One available only to the player who is misrepresenting and attempting to deceive.

To model this, we will use the framework of hypergame theory described in [40]. In hypergame theory, we assume that players involved are not seeing the same view of the game. In our case, the defender assumes that the attacker is continuing with the same rationality that it played during the learning stage when in reality, the attacker’s rationality has changed. This change in rationality allows the attacker to improve its expected utility without the defender’s awareness. This gives the attacker an extended view of the game while the defender gets only a partial perceived view of the game.

5.2 Hypergame Model

In this thesis, we assume the attacker has the option of using different rationalities between the learning and game stages. In the learning stage, the attacker can use a rationality level of α_{a_L}. Then use α_{a_A}, which is its actual rationality, during the actual game stage such that $\alpha_{a_L} < \alpha_{a_A}$, as the attacker is attempting to "play dumb". "Playing dumb" means that the attacker is trying to misrepresent themselves in front of the defender and is playing a lower rationality than their intended rationality. This way, the defender is expecting a lower rationality for the entirety of the game. However, once the learning stage is complete, the attacker changes their rationality to their actual intended rationality, deceiving the defender in the process.

Here, we model the game as a first level hypergame [22] such that the players have different views of the game. In the basic game, both players are assumed to have the same level of information. Thus, they have the same view of the game. However, under the considered deception, only the attacker has the option to deceive the defender, and the defender is assumed not to be aware of this deception. Therefore, the attacker will have the complete view of the defender's strategies while the defender has a limited view of the attacker's strategies. Let G_d be the defender's view of the attacker's strategy. This view will be as follows:

$$G_d = \begin{cases} \alpha_{a_A}, & \text{under no deception }, \\ \alpha_{a_L}, & \text{under deception}, \end{cases} \tag{5.1}$$

such that under a normal game without deception, the defender will perceive the attacker's actual rationality, which will be the same during both the learning phase and the actual game phase, i.e., α_{a_A}. However, under the deception case, the defender will only perceive the attacker's rationality during the learning phase, and it will not

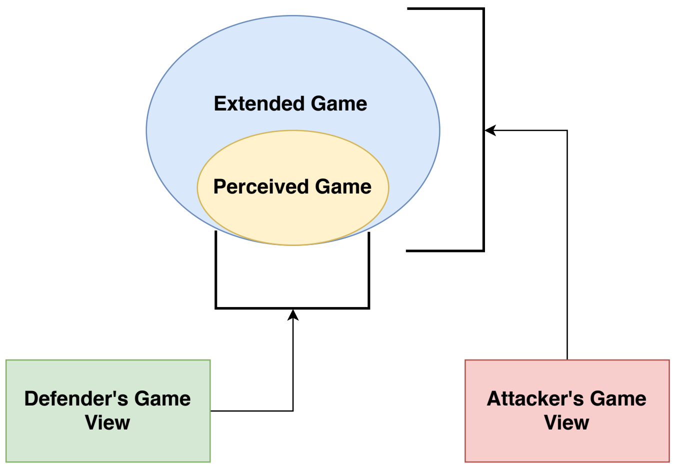

Figure 5.1: The attacker's and the defender's different views of the game

be aware of the rationality change in the actual game. Similarly, the attacker's view of the game can be given by G_a, which will equal the defender's actual rationality α_d, i.e., $G_a = \{\alpha_d\}$. The hypergame $\mathcal{H}$ can then be given by all players' views of the game, i.e., $\mathcal{H} = \{G_d, G_a\}$.

Since the defender is unaware of the deception, its utility in the game will be given by (4.7). On the other hand, the attacker, as being the deceiver, will have an extended view of the game which represents its outcome from the deception process. These different views of the same game are depicted in Fig. 5.1. The expected utility of the attacker, due to deception, can then be given as the difference between its utilities in the actual game phase with and without deception as follows:

$$\begin{aligned} U_a^H = C_A(&U_a^{PT}(p_a, p_d, \alpha_{a_A}, \alpha_d) \\ &- U_a^{PT}(p_a, p_d, \alpha_{a_L}, \alpha_d)). \end{aligned} \tag{5.2}$$

Equation (5.2) highlights the deception taking place in this game. After the learning stage, the defender is expecting the attacker to play the strategic profile that corresponds to the rationality level α_{a_L}. However, instead the attacker is playing with its actual rationality , α_{a_A}, which is a higher rationality. The defender only gets to see a partial view of the entire game, which is the game defined in Section 4.2. Thus, the defender will check for the trojans corresponding to the probability distributions of the attacker when its rationality is α_{a_L}. In fact, this utility will not reflect the actual status of the game as the attacker will be able to insert other trojans without being detected. The attacker, on the other hand, wants to maximize its utility, which is unknown to the defender. Based on its actual rationality α_{a_A}, the attacker wants to choose the deception rationality , α_{a_L}, from a set of rationalities $\mathcal{A}_\alpha$ under which its utility in (5.2) will be maximized. This can be done by solving the following optimization problem:

$$\begin{aligned} \operatorname*{arg\,max}_{\alpha_{a_L} \in \mathcal{A}_\alpha} \quad & C_A(U_a^{PT}(p_a, p_d, \alpha_{a_A}, \alpha_d) \\ & - U_a^{PT}(p_a, p_d, \alpha_{a_L}, \alpha_d)), \end{aligned} \tag{5.3}$$

which is equivalent to solving the difference between the two utilities:

$$\operatorname*{arg\,max}_{\alpha_{a_L} \in \mathcal{A}_\alpha} \quad U_a^{PT}(p_a, p_d, \alpha_{a_A}, \alpha_d) - U_a^{PT}(p_a, p_d, \alpha_{a_L}, \alpha_d). \tag{5.4}$$

Substituting (4.4) into (5.4), we get:

$$\begin{aligned} \operatorname*{arg\,max}_{\alpha_{a_L} \in \mathcal{A}_\alpha} \quad & \Big(p_d(s_d) w_d\big(p_a(s_a)|_{\alpha_{a_L}}, \alpha_d\big)\Big) u_a(s_d, s_a) \\ & - \Big(p_d(s_d) w_d\big(p_a(s_a)|_{\alpha_{a_A}}, \alpha_d\big)\Big) u_a(s_d, s_a), \end{aligned} \tag{5.5}$$

which can be simplified by omitting the common terms as:

$$\underset{\alpha_{a_L} \in \mathcal{A}_\alpha}{\arg\max}\ e^{(-(-p_a(s_a)|_{\alpha_{a_L}})^{\alpha_d})} - e^{(-(-p_a(s_a)|_{\alpha_{a_A}})^{\alpha_d})}, \tag{5.6}$$

which can be further simplified as:

$$\underset{\alpha_{a_L} \in \mathcal{A}_\alpha}{\arg\max}\ \ e^{(-p_a(s_a)|_{\alpha_{a_A}})^{\alpha_d} - (-p_a(s_a)|_{\alpha_{a_L}})^{\alpha_d}}. \tag{5.7}$$

From (5.7), we can see that an attacker can maximize its utility by choosing a rationality level that maximizes the difference between its probability distribution under this rationality level and the probability distribution under its actual rationality level when both probability distributions are weighted with the defender's rationality level.

Proposition 1 *The Nash equilibrium of the game, $\mathcal{G}$, along with the solution of* (5.7) *constitute a hyper Nash equilibrium to the game, $\mathcal{H}$.*

A strategy profile represents a hyper Nash equilibrium iff it belongs to the Nash equilibrium profile for each player's perceived game [41]. Since the defender's perceived game is only $\mathcal{G}$, the defender will have the same Nash equilibrium in $\mathcal{H}$ as $\mathcal{G}$. On the other hand, the solution to (5.7) represents the attacker's optimal solution based on its perceived game. Applying this solution to $\mathcal{G}$ will result in an equilibrium strategy for the attacker, for it will represent its best outcome based on its perceived game.

Finally, we summarize the steps of our deception mechanism in Fig. 5.2.

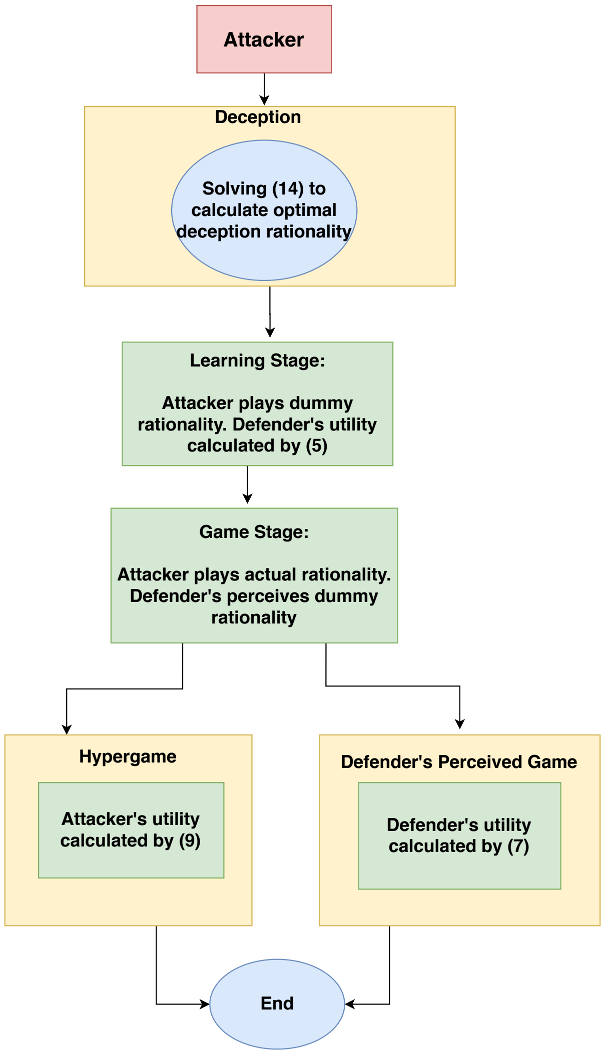
Attacker
Deception
Solving (14) to calculate optimal deception rationality
Learning Stage:
Attacker plays dummy rationality. Defender's utility calculated by (5)
Game Stage:
Attacker plays actual rationality. Defender's perceives dummy rationality
Hypergame
Attacker's utility calculated by (9)
Defender's Perceived Game
Defender's utility calculated by (7)
End

Figure 5.2: Flowchart of the framework

5.3 Deception Mitigation

The discrepancy between the expected utility of the attacker and its actual utility after deception is a massive incentive for the attacker to continue its deception. Without a proper deception countermeasure, the attacker will continue to deceive by using the same deception strategies on all subsequent batches to achieve higher payoffs. In this proposed scenario, as the defender is unaware of the deception, it needs some mechanism to mitigate the impacts of any potential deception, if any, in the actual game stage.

Here, we propose that the defender can run the learning stage again during the actual game stage to update its beliefs about the opponent's rationality. Since the attacker will be unaware of this update, it will keep playing the same strategies over and over, and it will incur losses as the defender will be able to detect the trojans. Note that, the premise behind the learning stage is to check every single integrated circuit (IC) in the batch to form an accurate belief about the attacker's strategies. Since this will consume a lot of time and will delay the production stage, the defender will be limited by the number of times it can re-run the learning stage. The defender's decision to run the learning stage again will then be based on the available resources and the time available for successful completion of checking. Let T be the total time by which all the batches need to be checked and delivered to the next production stage. Let T_L be the time taken to perform the learning stage on a single batch of ICs. Similarly, let T_A be the time taken to perform the actual game stage, i.e., checking C_A ICs on a single batch of ICs. Similar to 4.2, N is the total number of batches that are being tested. Let N_L be the number of batches that the defender will dedicate to the learning stages. Let N_A be the number of actual game batches, such that $N_A = N - N_L$. The maximum number of learning batches can then be given by solving the inequality:

$$T_L \cdot N_L + T_A \cdot N_A \leq T, \tag{5.8}$$

such that the total time used for both the learning and the actual game stages is less than or equal to T. From (5.8), the value of N_L can be given as:

$$\begin{aligned} N_L &\leq \frac{T - T_A \cdot N}{T_L - T_A}, \\ N_L &= \left\lfloor \frac{T - T_A \cdot N}{T_L - T_A} \right\rfloor . \end{aligned} \tag{5.9}$$

Depending on the value of N_L, the defender can distribute its learning stages uniformly during the extent of this game. This will allow the defender to not be completely dependent on the initial learning stage, but instead, it will allow them to run this learning stage multiple times during the course of the entire game in order to avoid the possibility of being deceived.

Note that, once the defender repeats its learning stage and takes new actions, the game theory analysis, discussed earlier, will not hold. This is because the attacker will not be able to observe the defender's actions nor take actions itself.

One way to model this situation is by using higher or additional levels of hypergame theory in which the defender is aware there is a different game being played [22] or that the defender realizes the attacker is being deceptive. The attacker, in return, will not be aware of the deception mitigation, which creates a new view of the game available to the defender only. The equilibrium for all these various views of the game will need to be considered in detail. However, this requires its own analysis and is left for future work. Here, this mitigation technique can be seen as a first step towards thwarting the effects of deception. Other future directions, that can be explored, is to use the frameworks of game-theoretic moving target defense [42] and [43] to model the case where the defender can randomize its defense strategies based on the attacker's expected strategies.

Chapter 6

Simulation Results and Analysis

To simulate this game situation, the following discrete values were assigned to the essential variables. We assumed that the attacker has access to 4 kinds of trojans, so $\mathcal{T}$ = A, B, C, D. Then, the strategy space of the attacker $S_A = \mathcal{T} = \{$A,B,C,D$\}$. Every trojan type has a corresponding utility for a successful attack. In this game, the utilities for each trojan are as follows: $V_A = 1$, $V_B = 2$, $V_C = 4$, and $V_D = 12$. The severity for the utilities is directly proportional to the magnitude of damage that can be done with that corresponding trojan. For the defender, it is assumed that $K = 2$. The value of K can be altered based on adequate resources available to the defender in order to effectively test for hardware trojans. Based on the value of K, the defender's strategy space, S_D, consists of $\binom{4}{2} = 6$ possible testing strategies. Then, $S_D = \{$AB, AC, AD, BC, BD, CD$\}$. The attacker's fine $F_{S_A} = F, \forall S_a \in S_A$. Here, we let $F = [8,6,2,4]$ to signify the penalty for the attacker on successful detection.

As this game is modeled in the fashion of a zero-sum game, Table 6.1 accurately portrays the desired static game scenario. Here, we can see the expected utilities for the attacker and the defender on successful detection or successful attack. When the

Table 6.1: Static Game Table

	Defender						
Att-acker		AB	AC	AD	BC	BD	CD
	A	-8,8	-8,8	-8,8	1,-1	1,-1	1,-1
	B	-6,6	2,-2	2,-2	-6,6	-6,6	2,-2
	C	4,-4	-2,2	4,-4	-2,2	4,-4	-2,2
	D	12,-12	12,-12	-4,4	12,-12	-4,4	-4,4

attacker chooses a trojan and the defender does not check for this specific trojan, the attacker's outcome will be positive while the defender's outcome will be negative and vice versa. Since the outcomes are alternating positively and negatively for each player based on the opponent's selection, there is no dominant strategy for any player.

To initiate the simulation, we start by executing the fictitious play algorithm, which requires initializing the strategic probabilities of both players. Here, we use the same initial probabilities as in [12]. For the attacker, the initial strategic profile $p_a =$ [0.2083, 0.1667, 0.3333, 0.2917] and for the defender, the initial strategic profile $p_d =$ [0.2051, 0.2564, 0.2564, 0.0513, 0.0513, 0.1795].

To study the effect of these initialization probabilities on the final equilibrium results in Fig. 6.1, we show the effect of different initialization probabilities on final equilibrium probabilities. Fig. 6.1 highlights the attacker's probability of playing trojan D under six different attacker's initialization mixed strategy probability vectors. Note that, we only show the probability of choosing trojan D as it turned out to be the attacker's preferred strategy when it has a rationality of 0.3. Also, each curve in Fig. 6.1 starts at a different initialization probability while the attacker's rationality is fixed for all the initialization vectors at 0.3. Fig. 6.1 shows the probability of choosing trojan D as it evolves over the steps of executing the fictitious play algorithm. We can see that the initial mixed strategy probability distributions do not affect the final equilibrium probability as all the curves converge to the same value. However,

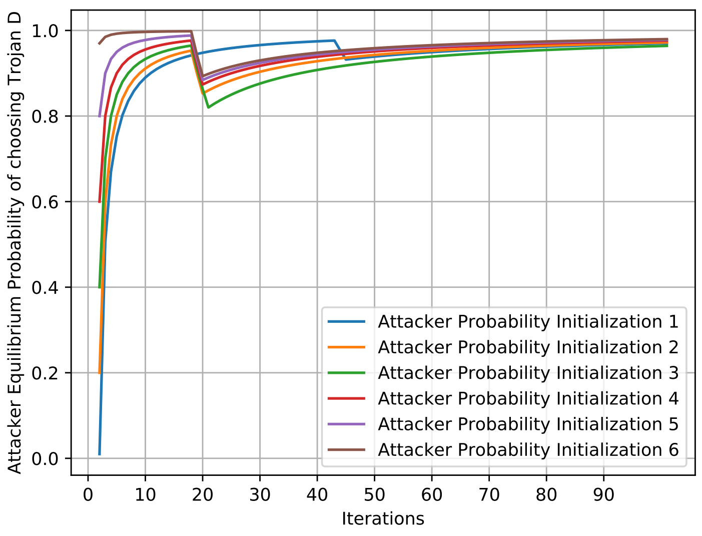

Figure 6.1: Rate of Convergence for Attacker Equilibrium Probability of choosing Trojan D

the effect of the initialization probabilities can be seen in the number of iterations needed for the convergence to occur, which varies from one initialization probability to another. We can then conclude that no matter which initialization probabilities are used in the following results, the equilibrium mixed strategy probability vectors will always be the same at convergence.

For the attacker, we let its actual rationality, α_{a_A}, to be 0.5. Then, we apply the deception problem in (5.7) to compute the deception rationality, i.e., the rationality the attacker will use in the actual game. In this case, it was shown that the attacker can maximize its utility in the hypergame if it uses the learning dummy rationality of $\alpha_{a_L} = 0.1$. Therefore, the attacker will set $\alpha_{a_L} = 0.1$.

Then, to study the utilities of both players in the learning stage, we run the fictitious play algorithm when $\alpha_d = 0.5$ and when α_{a_L} equals both 0.1 and 0.5. Fig. 6.2 shows the defender's strategic profile, at equilibrium, for both the the attacker's rationalities. These probability distributions represent the defender's probability of selecting each of its potential strategies in S_D. From Fig. 6.2, we can see that when $\alpha_{a_L} = 0.1$, the

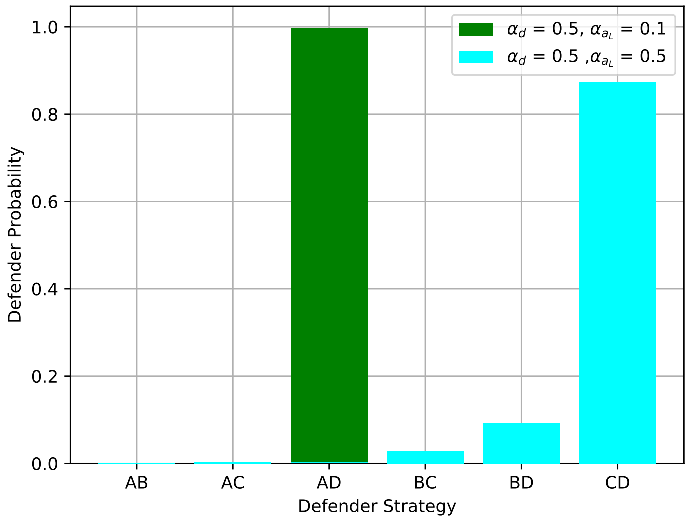

Figure 6.2: Defender strategic profile when the defender's rationality $\alpha_d = 0.5$ against attacker's rationalities of $\alpha_{a_L} = 0.1$ and $\alpha_{a_L} = 0.5$.

defender will choose the strategy AD with high probability. However, this changes when the attacker's rationality changes to $\alpha_{a_L} = 0.5$ as the defender will have a more distributed probability vector in which it will choose strategy CD with a probability around 0.87 and use other strategies with some small probabilities.

Similarly, Fig. 6.3 shows the attacker's strategic profile when $\alpha_d = 0.5$ and when α_{a_L} equals both 0.1 and 0.5. We can see that for $\alpha_{a_L} = 0.1$, the attacker chooses the strategy D with high probability. This corroborates with the defender's choice of AD with high probability as the strategy D is common between both player's strategies. Similarly, when $\alpha_{a_L} = 0.5$, the attacker has a more distributed probability vector in which it will choose strategy C with a probability of around 0.5 and other strategies with smaller probabilities.

We then study the players' utilities based on the equilibrium strategies shown in Figs. 6.2 and 6.3. These utilities are calculated using (4.7) where each player will have the exact same utility for each batch of the game, and their total utility will be the batch utility multiplied by the number of the batches. Therefore, in Fig. 6.4, we show both

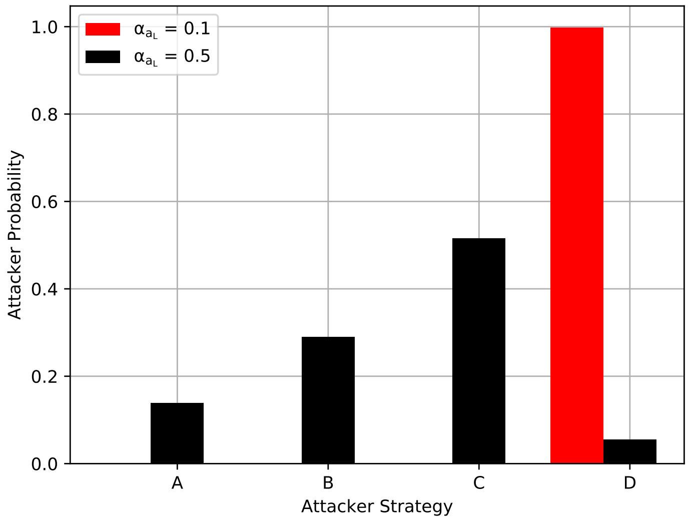

Figure 6.3: Attacker strategic profile when the defender's rationality $\alpha_d = 0.5$ against attacker's rationalities of $\alpha_{a_L} = 0.1$ and $\alpha_{a_L} = 0.5$.

players' utilities for a single batch.

Fig. 6.4 shows, that when the attacker has a rationality of $\alpha_{a_L} = 0.1$ in both the learning stage and the actual game stage and the defender has a rationality of $\alpha_d = 0.5$, the attacker incurs a utility hit of -3.9614854 per IC. Meanwhile, the defender receives a utility of 3.9614854 per IC. In retrospect, when the attacker plays a rationality of $\alpha_{a_L} = 0.5$ in both the learning stage and the actual game stage, the attacker receives a utility hit of -0.4984252 per IC. The zero-sum nature continues, and the defender receives a utility of 0.4984252 per IC. This shows that the attacker will incur a negative outcome when it plays the same rationality in both the game stages, no matter whether it played high or low rationalities. This is because the defender randomizes over its actions, and it will be able to detect the hardware trojans with high probability.

Next, we study the effect of deception to the attacker's utility. In this case, during the learning stage of the game, the attacker decides to deceive the defender by playing a lower rationality level, α_{a_L}. The defender will receive the attacker's probability

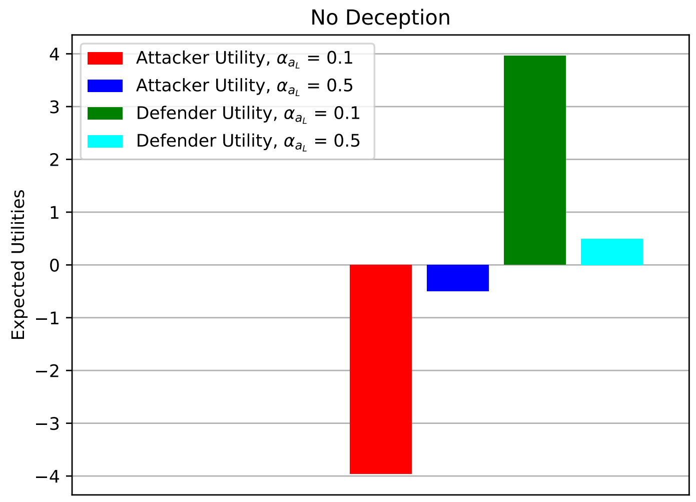

Figure 6.4: The attacker's and the defender's utilities under no deception when $\alpha_{a_L} = \alpha_{a_A}$.

distribution corresponding to $\alpha_{a_L} = 0.1$ in Fig. 6.3, and thus, will be playing the corresponding profile from Fig. 6.2. This means the defender will mainly be adopting the strategy AD. However, in the actual game stage, the attacker will change its strategy as it will play its actual rationality of $\alpha_{a_A} = 0.5$.

The outcome of this deception case is shown in Fig. 6.5. From Fig. 6.5, we can see that when the attacker plays $\alpha_{a_L} = 0.1$ during learning and then $\alpha_{a_A} = 0.5$ during the actual game stage, while the defender is playing $\alpha_d = 0.5$ assuming the attacker is playing $\alpha_{a_A} = 0.1$, the defender receives a utility hit of -1.3090019 per integrated circuit (IC) while the attacker receives a utility gain of 1.3090019 per IC. The previous no deception case is also included in Fig. 6.4 for reference, in which the attacker gets detected and paying a fine for the defender.

Note that, while Fig. 6.5 shows the defender with a negative utility, the defender in fact will not be aware of these losses. From a practical point of view, the defender will focus on the profile AD, as discussed earlier, and it will not detect most of the trojans. Thus, it will lose the utility corresponding to each undetected trojan. Similarly, the attacker will gain this value for every single IC in each batch of the actual game stage.

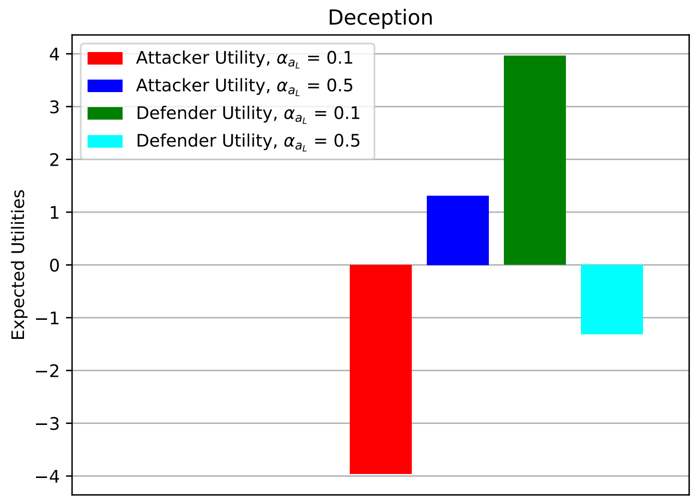

Figure 6.5: The attacker's and the defender's utilities under deception when $\alpha_{a_L} \neq \alpha_{a_A}$ and under no deception when $\alpha_{a_L} = \alpha_{a_A}$.

The accumulated utility of the attacker from the hypergame model, as defined in (5.2), will , however, be different. This utility represents the goal of the attacker of maximizing its utility between playing its actual rationality and playing “dumb" in the learning stage. Since this utility depends both on the attacker's deception rationality and its actual rationality, in Fig. 6.6, we study different scenarios for the attacker's original rationality and how they affect its accumulated utility. Moreover, we study these cases for different defender's perceptions, i.e., defenders with different rationalities.

From Fig. 6.6, we can see that irrespective of the attacker's actual rationality, the attacker's utility will always be higher when faced with a defender with a lower rationality. For instance, when the attacker's actual rationality is 0.5, its highest deception utility gain is when the defender's rationality is 0.5. Similarly, when the attacker's actual rationality of 0.6, its highest deception utility gain is when it faces defenders of rationalities, 0.5 and 0.6. However, if the defender's rationality is higher than 0.6 , the attacker will still achieve a positive utility, but the order becomes less predictable. For instance, when the attacker's actual rationality is 0.5, its utility

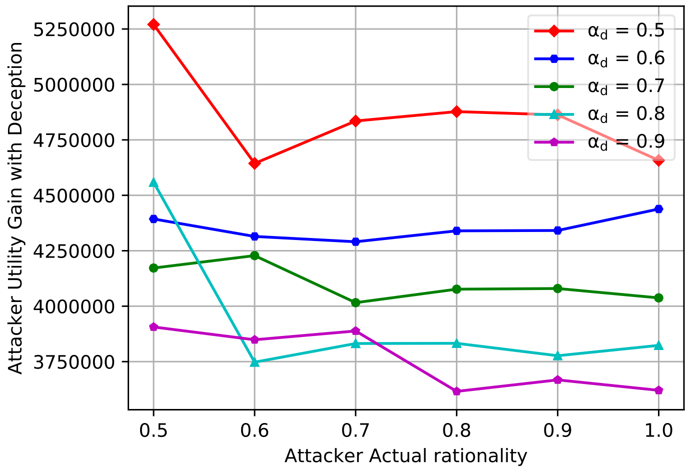

Figure 6.6: Attacker's utility gain under different defender's rationalities when the optimal deception rationality is played for each type.

will be higher when it faces a defender with the rationality of 0.8, than when it faces a defender with the rationality of 0.6. The same happens for the attacker's rationalities of 0.6 and 0.7, as the attacker achieves a higher utility when the defender has rationality of 0.9 compared to when the defender has a rationality of 0.8.

Another interesting finding in Fig. 6.6 is that an attacker with a higher actual rationality does not have to gain more than an attacker with a lower rationality. For instance, when facing a defender with a rationality of 0.7, the best possible utility will be achieved by an attacker with an actual rationality of 0.6. Similarly, against a defender with a rationality of 0.6 , the best possible utility will be achieved by an attacker with a rationality of 1.0. This, in fact, corroborates the importance of hypergame theory as the maximum achievable utility for the attacker depends on the deception rationality, which may differ for each actual rationality.

Finally, Fig. 6.7 studies the utility of different attacker's, based on their actual rationalities α_{a_A}. Fig. 6.7 shows the utility of each attacker's type based on their choice of the deception rationalities, α_{a_L}. Each curve represents a type of attacker, and it extends from the lower possible rationality up to the attacker's actual rationality,

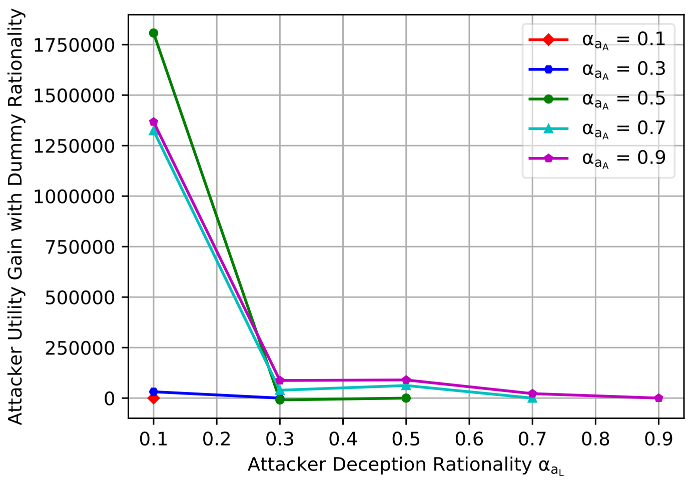

Figure 6.7: The utility gain of each attacker's type when they play different deception rationalities α_{a_L} in the learning stage.

for an attacker cannot play a higher rationality than its actual rationality. We can see that the deception rationality of $\alpha_{a_L} = 0.1$ achieves the highest utility of any attacker when its actual rationality is greater than 0.3. Another important finding from Fig. 6.7, is that an attacker with a lower rationality, for e.g., 0.3 will not be able to benefit well from the deception as its utility will remain comparatively very small. Similarly, an attacker with high rationality will maximize its utility if it plays a deception rationality lower than or equal to 0.3. This corroborates the findings from Fig. 4.1 about the inflection point of 0.37 and its effect on flipping the order of a player's strategies. Note, in Fig. 6.7, the optimal deception rationality for all the players is 0.1; however, this is not a general case in deception scenarios, and the solution of (5.7) should be used to compute the optimal deception rationality based on each game's parameters.

Chapter 7

Conclusion and Future Work

In this thesis, we have proposed a novel framework for deception in hardware trojan detection systems. Prospect theory has been used to model the basic players' utilities, without deception, in order to account for different players' rationalities. These rationalities represent the nature of human subjectivity when faced with uncertainty. We then formulated a repeated game in which the defender learns about the attacker's strategies in the learning stage and then applies this learning knowledge to the subsequent actual game stage of the game. The incentive behind deception has been carefully discussed, which is built on the premise of the Prelec function's effect on flipping the order of evaluation between low and high probabilities. Thus, an attacker will find it motivating to use different rationality levels in both the learning and the actual game stage of the repeated game. We have then formulated an extended view of the game using a hypergame model in which the attacker has a complete view of the game while the defender has a partial perceived view of the game. The hypergame model allows the attacker to optimally determine its deceiving rationality level to maximize its utility. We have also proposed a first-step defense against the deception by allowing the defender, while resources permit, to repeat the learning stage in order to mitigate the effects of deception. Finally, we have tested the proposed framework using simulations. The results have shown that the attacker can successfully insert hardware trojans without being detected. Results have also shown the attacker's gain in utility under different combinations of rationalities of the attacker and the defender. For future work, we will focus on building more rigor defense mechanisms using moving target defense and higher levels of hypergame theory.

One promising direction for future work is to study the deception game when the attacker has the option of "no trojan", i.e., it chooses not to insert a trojan. Similarly, the strategy of "no test" can be considered for the defender when it chooses not to test the current integrated circuit for hardware trojans. These strategies were considered in [16] for the static game scenario, and they represent a good extension for the hypergame considered in this thesis. Lastly, another potential research extension on this research can be analyzing the game scenario, assuming the attacker can insert more than one trojan per trojan.

Bibliography

[1] M. Haselman and S. Hauck, "The future of integrated circuits: A survey of nanoelectronics," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 11–38, jan 2010.

[2] B. N. Hwang, T. T. Chen, and J. T. Lin, "3PL selection criteria in integrated circuit manufacturing industry in Taiwan," *Supply Chain Management*, vol. 21, no. 1, pp. 103–124, 2016.

[3] J. Dofe, Q. Yu, H. Wang, and E. Salman, "Hardware security threats and potential countermeasures in emerging 3D ICs," in *Proceedings of the ACM Great Lakes Symposium on VLSI, GLSVLSI*, 2016.

[4] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *Proceedings - IEEE International High-Level Design Validation and Test Workshop, HLDVT*, 2009.

[5] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, October 2010.

[6] T. Boraten and A. K. Kodi, "Mitigation of denial of service attack with hardware trojans in noc architectures," in *2016 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE, 2016, pp. 1091–1100.

[7] S. Sengupta, K. Hong, R. Chandramouli, and K. P. Subbalakshmi, "Spiderradio: A cognitive radio network with commodity hardware and open source software," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 101–109, 2011.

[8] A. Eldosouky and W. Saad, "On the cybersecurity of m-health iot systems with led bitslice implementation," in *2018 IEEE International Conference on Consumer Electronics (ICCE).* IEEE, 2018, pp. 1–6.

[9] T. Brodeur, P. Regis, D. Feil-Seifer, and S. Sengupta, "Search and rescue operations with mesh networked robots," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON).* IEEE, 2018, pp. 6–12.

[10] A. N. Patra, P. A. Regis, and S. Sengupta, "Dynamic self-reconfiguration of unmanned aerial vehicles to serve overloaded hotspot cells," *Computers & Electrical Engineering*, vol. 75, pp. 77–89, 2019.

[11] A. French, M. Mozaffari, A. Eldosouky, and W. Saad, "Environment-aware deployment of wireless drones base stations with google earth simulator," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops).* IEEE, 2019, pp. 868–873.

[12] W. Saad, A. Sanjab, Y. Wang, C. A. Kamhoua, and K. A. Kwiat, "Hardware Trojan Detection Game: A Prospect-Theoretic Approach," *IEEE Transactions on Vehicular Technology*, 2017.

[13] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic, "Hardware Trojan detection and isolation using current integration and localized current analysis," in *Proceedings - IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2008.

[14] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game theory in wireless and communication networks: theory, models, and applications.* Cambridge university press, 2012.

[15] J. Graf, "Trust games: How game theory can guide the development of hardware Trojan detection methods," in *Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016*, 2016.

[16] C. A. Kamhoua, H. Zhao, M. Rodriguez, and K. A. Kwiat, "A Game-Theoretic Approach for Testing for Hardware Trojans," *IEEE Transactions on Multi-Scale Computing Systems*, 2016.

[17] X. D. He and X. Y. Zhou, "Portfolio choice under cumulative prospect theory: An analytical treatment," *Management Science*, 2011.

[18] N. C. Barberis, "Thirty years of prospect theory in economics: A review and assessment," 2013.

[19] A. Sanjab, W. Saad, and T. Başar, "Prospect theory for enhanced cyber-physical security of drone delivery systems: A network interdiction game," in *2017 IEEE International Conference on Communications (ICC).* IEEE, 2017, pp. 1–6.

[20] L. Xiao, N. B. Mandayam, and H. Vincent Poor, "Prospect theoretic analysis of energy exchange among microgrids," *IEEE Transactions on Smart Grid*, 2015.

[21] E. S. Al-Shaer, J. Wei, K. W. Hamlen, and C. Wang, *Autonomous Cyber Deception: Reasoning, Adaptive Planning, and Evaluation of HoneyThings.* Springer, 2019.

[22] N. S. Kovach, A. S. Gibson, and G. B. Lamont, "Hypergame theory: a model for conflict, misperception, and deception," *Game Theory*, vol. 2015, 2015.

[23] D. Sklansky, *The theory of poker.* Two Plus Two Publishing LLC, 1999.

[24] A. Petrovic, I. Markovic, V. Koprivica, and B. Bokan, "Tactics factors in chess: Theoretical-empirical aspects."

[25] E. Rasmusen, *Games and information: An introduction to game theory.* Blackwell Oxford, 1989, no. 519.3/R22g.

[26] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.

[27] D. Jiang and J. Hu, "Research of key technology in game theory," in *2010 The 2nd Conference on Environmental Science and Information Application Technology*, vol. 2. IEEE, 2010, pp. 639–642.

[28] V. N. Kolokoltsov and O. Malafeyev, "Understanding game theory: introduction to the analysis of many agent systems with competition and cooperation," *UNDERSTANDING GAME THEORY*, 2010.

[29] A. M. Colman, *Game theory and its applications: In the social and biological sciences.* Psychology Press, 2013.

[30] D. E. Charilas and A. D. Panagopoulos, "A survey on game theory applications in wireless networks," *Computer Networks*, vol. 54, no. 18, pp. 3421–3430, 2010.

[31] Y. Shoham, "Computer science and game theory," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2008.

[32] O. Morgenstern and J. Von Neumann, *Theory of games and economic behavior.* Princeton university press, 1953.

[33] O. Chatain, *Cooperative and Non-cooperative Game Theory.* London: Palgrave Macmillan UK, 2016, pp. 1–3. [Online]. Available: https://doi.org/10.1057/978-1-349-94848-2_468-1

[34] S. Sengupta, "An economic framework for resource management and pricing in wireless networks with competitive service providers," 2007.

[35] P. Bajari, H. Hong, and S. P. Ryan, "Identification and estimation of a discrete game of complete information," *Econometrica*, vol. 78, no. 5, pp. 1529–1568, 2010.

[36] X. Deng and J. Deng, "A study of prisoner's dilemma game model with incomplete information," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[37] J. O. Neel, J. H. Reed, and R. P. Gilles, "Convergence of cognitive radio networks," in *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No. 04TH8733)*, vol. 4. IEEE, 2004, pp. 2250–2255.

[38] D. Prelec, "The Probability Weighting Function," *Econometrica*, vol. 66, no. 3, p. 497, 1998.

[39] S. Sengupta, M. Chatterjee, and K. Kwiat, "A game theoretic framework for power control in wireless sensor networks," *IEEE Transactions on Computers*, vol. 59, no. 2, pp. 231–242, 2009.

[40] P. G. Bennett, "Hypergames: Developing a model of conflict," *Futures*, 1980.

[41] Y. Sasaki, N. Kobayashi, and K. Kijima, "Mixed extension of hypergames and its applications to inspection games," in *Proceedings of the 51st Annual Meeting of the ISSS-2007, Tokyo, Japan*, vol. 51, no. 2, 2007.

[42] Q. Zhu and T. Başar, "Game-theoretic approach to feedback-driven multi-stage moving target defense," in *International Conference on Decision and Game Theory for Security*. Springer, 2013, pp. 246–263.

[43] A. Eldosouky, W. Saad, and D. Niyato, "Single controller stochastic games for optimized moving target defense," in *2016 IEEE International Conference on*

Communications (ICC). IEEE, 2016, pp. 1–6.