Strategic Remote Attestation: Testbed for Internet-of-Things Devices and Stackelberg Security Game for Optimal Strategies

Shanto Roy[1], Salah Uddin Kadir[1], Yevgeniy Vorobeychik[2], and Aron Laszka[1]

[1] University of Houston, Houston, TX
[2] Washington University in St. Louis, St. Louis, MO

Published in the proceedings of the 12th Conference on Decision and Game Theory for Security (GameSec 2021).

Abstract. Internet of Things (IoT) devices and applications can have significant vulnerabilities, which may be exploited by adversaries to cause considerable harm. An important approach for mitigating this threat is *remote attestation*, which enables the defender to remotely verify the integrity of devices and their software. There are a number of approaches for remote attestation, and each has its unique advantages and disadvantages in terms of detection accuracy and computational cost. Further, an attestation method may be applied in multiple ways, such as various levels of software coverage. Therefore, to minimize both security risks and computational overhead, defenders need to decide strategically which attestation methods to apply and how to apply them, depending on the characteristic of the devices and the potential losses.
To answer these questions, we first develop a *testbed for remote attestation of IoT devices*, which enables us to measure the detection accuracy and performance overhead of various attestation methods. Our testbed integrates two example IoT applications, memory-checksum based attestation, and a variety of software vulnerabilities that allow adversaries to inject arbitrary code into running applications. Second, we model the problem of finding an optimal strategy for applying remote attestation as a *Stackelberg security game* between a defender and an adversary. We characterize the defender's optimal attestation strategy in a variety of special cases. Finally, building on experimental results from our testbed, we evaluate our model and show that optimal strategic attestation can lead to significantly lower losses than naïve baseline strategies.

Keywords: Remote Attestation · Stackelberg Security Game · Internet of Things · Security Testbed · Software Security

1 Introduction

With the growing number of Internet of Things (IoT) devices around the world, security has been a significant concern for researchers in the last decade. Due to

more exposure in a resource-limited environment, IoT devices often do not have access to the latest security primitives, and a number of security issues including various software vulnerabilities (e.g., stack and heap-based buffer overflows, format-string vulnerabilities) exist due to the usage of unsafe languages like C/C++ and vulnerable functions [21,17,11]. Adversaries can exploit these vulnerabilities to compromise devices by altering the software code or the control flow. Therefore, from a defensive point of view, device attestation that allows an organization to verify integrity remotely is a powerful tool [17].

IoT devices are a preferable target for adversaries these days, and organizations are implementing various methods to mitigate these attacks. However, security measures for IoT devices are different from those for servers, since IoT devices usually have low-power resource-limited configurations and are often placed in unknown or unsafe locations. To detect and mitigate attacks, a defender may employ remote attestation methods to verify the integrity of a program.

While remote attestation methods can be effective at detecting compromised devices, running attestation can also incur significant computational cost, which may present a prohibitively high overhead on resource-limited IoT devices. There are a number of approaches for remote attestation, and each has its unique advantages and disadvantages in terms of detection accuracy and computational cost. Further, an attestation method may be applied in multiple ways, such as various levels of software coverage. Therefore, to minimize both security risks and computational overhead, defenders need to decide strategically which attestation methods to apply and how to apply them, depending on the characteristic of the devices and the potential losses.

In this paper, we address these questions by (1) implementing an IoT testbed for measuring the detection accuracy and performance overhead of remote-attestation methods and by (2) introducing and solving a game-theoretic model for finding optimal remote-attestation strategies. Specifically, we formulate and answer the following research questions.

Q1. Testbed Development: How to develop an IoT security testbed that can simulate software vulnerability exploitation and evaluate remote attestation?
Q2. Remote Attestation Methods: What is the trade-off between the detection rate and computational cost of various remote attestation methods?
Q3. Optimal Attestation Strategies: How to model the strategic conflict between a defender and an adversary, and how to find optimal attestation strategies for the defender?

We answer the first question by describing the design and development of our security testbed for IoT device attestation (Section 3). We discuss the architecture of our testbed as well as the development of major components, such as vulnerability exploits and attestation methods. Our testbed enables us to experiment with software vulnerabilities and exploits and to rigorously evaluate various attestation methods in terms of computational cost and detection rate.

We answer the second question by studying the detection rate and computational cost of memory-checksum based remote attestation (Section 6). We implement and evaluate memory-checksum based attestation in our testbed for

two example IoT applications. We characterize the trade-off between computational cost and detection rate, which we then use to develop the assumptions of our game-theoretic model.

We answer the third question by developing a Stackelberg security game to model the strategic conflict between a defender and an adversary (Section 4). We formulate the defender's optimal remote-attestation strategy assuming an adversary who always mounts a best-response attack. We show how to compute an optimal strategy in various special cases, and we demonstrate through numerical examples that optimal strategies can attain significantly lower losses than naïve baselines. To the best of our knowledge, our model and analysis constitute the first effort to provide optimal remote-attestation strategies.

Organization The rest of the paper is organized as follows: Section 2 provides necessary background information. Section 3 discusses the design and development details of our IoT security testbed. Section 4 introduces the attacker-defender model based on Stackelberg security games. Section 5 provides analytical results characterizing the defender's optimal attestation strategy. Section 6 presents experimental results from our testbed as well as numerical results on the optimal attestation strategies. Section 7 gives a brief overview of related work followed by our concluding remarks and future directions in Section 8.

2 Background

ARM processors are very widely used in IoT platforms. Therefore, we develop an ARM-based IoT security testbed to experiment with exploitation and remote attestation on ARM devices. Here, we provide a brief overview of IoT device vulnerabilities, remote attestation methods, and the Stackelberg game model.

2.1 Software Vulnerabilities and Exploitation in IoT Devices

Adversaries can take control of an IoT device by hijacking the code execution flow of an application and injecting arbitrary executable code into its memory space. For example, an attacker can use stack- or heap-based *buffer overflow* or *format string vulnerabilities* to inject malicious executable code into a process. By injecting executable code, the adversary can alter the functionality of an application (e.g., providing a backdoor to the adversary or causing harm directly). While the mitigation for these attacks may be well established in the server and desktop environment, the unique design characteristics of resource-constrained embedded devices makes it challenging to adapt the same defenses techniques. For example, many deeply embedded devices often do not support virtual memory, which is essential for address space layout randomization (ASLR).

2.2 IoT Remote Attestation

Remote attestation establishes trust in a device by remotely verifying the state of the device via checking the integrity of the software running on it. Remote

attestation methods can be divided into two main categories: hardware and software based. Hardware-based attestation requires additional dedicated hardware (e.g., Trusted Platform Module) on the device [1]. Deploying dedicated hardware can incur additional cost in terms of hardware cost and power consumption, which are often prohibitive for inexpensive or low-power devices. In contrast, software-based attestation requires a *software prover* on the device, which performs specific computations (e.g., memory- or time-based checksum [13,14]) and returns the result to the verifier. Note that there are also hardware-software co-design hybrid platforms for remote attestation [10]. In this paper, we focus on software-based remote attestation.

Steiner et al. categorized checksum-based memory attestation in terms of evidence acquisition (software-based, hardware-based, or hybrid), integrity measurement (static or dynamic), timing (loose or strict), memory traversal (sequential or cell/block-based pseudo random), attestation routine (embedded or on-the-fly), program memory (unfilled or filled), data memory (unverified, verified, or erased), and interaction pattern (one-to-one, one-to-many, or many-to-one) [17]. Memory checksums can be generated based on *sequential* or *pseudo-random* traversal. In sequential traversal, each program memory cell is accessed in a sequential order. In contrast, in pseudo-random traversal, memory is accessed in a random cell-by-cell or block-by-block order. The effectiveness of pseudo-random traversal depends on the probability that each cell has been accessed at least once.

2.3 Stackelberg Security Games

A Stackelberg security game (SSG) is a game-theoretic model, where typically a defending player acts as the leader, and the adversarial player acts as the follower. The leader has the advantage of making the first move, while the follower has the advantage of responding strategically to the leader's move. Stackelberg security games have been successfully applied to finding optimal defensive strategies in a variety of settings, both in the cyber and physical domain [16]. For example, SSG have helped researchers and practitioners to address a security issues such as security-resource allocation at airports, biodiversity protection, randomized inspections, road safety, border patrol, and so on [19,7,24,5].

Game theory can model attacker-defender interactions and characterize optimal strategies given the players' strategy spaces and objectives. In our game-theoretic model of remote attestation, the defender acts as the leader by deciding how often to perform remote attestation, and the adversary acts as the follower by deciding which devices to attack. We provide detailed definitions of the environment, the player's strategy spaces, and their objectives in Section 4.

3 Testbed Design and Development

In our testbed, multiple IoT applications are running on multiple IoT devices. We implement and enable various software vulnerabilities (e.g., heap-based buffer

overflow) in these applications so that adversaries can remotely compromise the devices by exploiting these vulnerabilities. As a result, adversaries can modify the code of processes without crashing or restarting them. We also integrate memory checksum-based attestation method in the applications. Therefore, a verifier can remotely verify the integrity of the vulnerable processes.

3.1 Testbed Components

A typical IoT testbed consists of several IoT devices running various IoT server applications. Our testbed also includes two other types of nodes to mount attacks (e.g., code injection) against the IoT devices and to detect the attacks using remote attestation. The architecture of our testbed is presented in Figure 1.

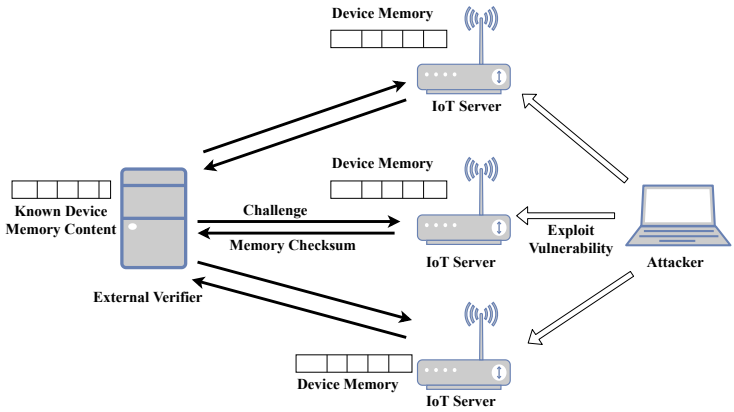


Fig. 1: Remote attestation testbed architecture.

IoT Server Node We set up various IoT server applications on these nodes, add vulnerable code snippets to the applications, and debug them to find exploitation opportunities that can be used to perform code-injection attacks. Then, we incorporate a memory-checksum generator that can calculate a checksum whenever the application receives a challenge from an external verifier node.

Attacker Node The attacker is a client node that can interact with the IoT application servers and execute various exploits (e.g., stack- or heap-based buffer overflow). The attacker's purpose is to inject or alter the software code of vulnerable applications without crashing the processes.

External Verifier Node The verifier is responsible for performing memory-checksum based attestation of the potentially compromised application servers. For checksum-based attestation, the verifier sends a challenge along with a random seed to the potentially compromised server node and records the response in return to verify.

3.2 Testbed Development

To experiment with various remote attestation strategies, we implement the following features in the testbed: start or terminate various IoT applications, exploit these applications from an attacker node, and generate challenge-response for remote attestation.

Testbed Setup Our testbed uses five Raspberry Pi 3 Model B+ devices (two IoT application servers, an attacker, and two nodes for the verifier). All devices run Raspbian Linux. We incorporate two example IoT applications: an irrigation server[3] and a smart home[4].

Enabling Vulnerabilities We disable certain security features of the Linux kernel and the compiler to enable stack- and heap-overflow based exploitation. To enable these vulnerabilities, we disable the ASLR and stack protection; and enable code execution while compiling the applications.

Exploitation Simulation We debug all of the applications on the application server nodes to find stack- and heap-based vulnerabilities. Then, we create corresponding exploit payloads on the attacker node. The attacker node sends a request to the server, which triggers the vulnerability and thereby injects a shellcode into the process.

Integrity Verification Simulation In our testbed, we implement memory-checksum (sequential- and random-order checksum) as remote attestation strategies, which require an external trusted verifier. The verifier can attest a potentially compromised device by sending the same challenge to the target device and an identical isolated device, and compare their responses.

4 Game-Theoretic Model of Remote Attestation

Remote attestation enables a defender to detect compromised devices remotely. However, the effectiveness and computational cost of attestation depends on strategies, such as when to attest a device and what method of attestation to employ. As IoT devices are resource-limited and attestation incurs computational cost, some devices should not be verified frequently (e.g., devices with low value for an adversary). On the other hand, some devices (e.g., ones with high value for the adversary) may need frequent attestation.

To find optimal strategies for remote attestation, we propose a game-theoretic model. Our model is a *two-player, general-sum Stackelberg security game*, where the defender is the leader and the adversary is the follower (i.e., defender first selects its attestation strategy to defend its IoT devices, and then the adversary chooses which devices to attack considering the defender's attestation strategy).

[3] `https://github.com/NamedP1ayer/IrrigationServer`
[4] `https://github.com/renair/smarthome`

We assume that the defender chooses for each device and for each attestation method the probability of applying that method to that device; the adversary chooses for each device whether to attack it or not.

Table 1 summarizes the notation of our game-theoretic model.

4.1 Environment & Players

There is a set of IoT devices $\mathcal{D}$ in the environment, where each individual device $\delta \in \mathcal{D}$ runs various IoT applications and services. As different devices may have different software stacks, we divide the devices into disjoint classes. These device classes are denoted $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$, where we have $i \neq j \rightarrow \mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ and $\bigcup_i \mathcal{E}_i = \mathcal{D}$. Devices belonging to the same class have the same vulnerabilities and may be attacked using the same exploits.

In our Stackelberg game model, there are two players: a defender (leader) and an attacker (follower). The defender tries to minimize the security risks of the IoT devices by detecting compromises, while the attacker tries to compromise the devices but avoid detection. To detect compromised devices, the defender uses various attestation methods (e.g., memory checksum, control-flow integrity). We let $\mathcal{M}$ denote the set of attestation methods, where each attestation method is an $m \in \mathcal{M}$. If the defender detects a compromised device, the defender resets the device back to its secure state.

4.2 Strategy Spaces

Knowing the defender's strategy (i.e., probability of attesting each device using each method), the attacker chooses which devices to attack. We assume that the attacker follows a deterministic strategy and chooses for each device whether to attack it or not. Note that in an SSG, restricting the follower (i.e., the attacker) to deterministic strategies is without loss of generality. We let the attacker's strategy be represented as a vector $\boldsymbol{a} = \langle a_\delta \rangle_{\delta \in \mathcal{D}}$, where $a_\delta = 1$ means attacking device $\delta \in \mathcal{D}$, and $a_\delta = 0$ means not attacking device δ. Therefore, the attacker's strategy space is

$$\boldsymbol{a} \in \{0, 1\}^{|\mathcal{D}|}.$$

On the other hand, the defender can choose a randomized strategy, i.e., for each device $\delta \in \mathcal{D}$ and attestation method $m \in \mathcal{M}$, the defender chooses the probability $p_\delta^m \in [0, 1]$ of running attestation method m on device δ. We let the defender's strategy be represented as a vector $\boldsymbol{p} = \langle p_\delta^m \rangle_{\delta \in \mathcal{D}, m \in \mathcal{M}}$, where $p_\delta^m = 0$ means never running method $m \in \mathcal{M}$ on device $\delta \in \mathcal{D}$, and $p_\delta^m = 1$ means always running method m on device δ. Therefore, the defender's strategy space is

$$\boldsymbol{p} \in [0, 1]^{|\mathcal{D} \times \mathcal{M}|}.$$

4.3 Utility Functions

Next, we formalize the players' objectives by defining their utility functions.

Table 1: List of Symbols

Symbol	Description
Constants	
$\mathcal{D}$	set of devices
$\mathcal{M}$	set of attestation method
$\mathcal{E}_i$	a set of devices that share common vulnerabilities, where $\mathcal{E}_i \subseteq \mathcal{D}$
μ^m	detection probability of attestation method $m \in \mathcal{M}$ when executed on a compromised device
C_D^m	defender's cost to run attestation method $m \in \mathcal{M}$
C_A^δ	attacker's cost to compromise device $\delta \in \mathcal{D}$
$C_A^\mathcal{E}$	attacker's cost to develop an exploit for a device class $\mathcal{E} \subseteq \mathcal{D}$
G_D^δ, G_A^δ	defender's / attacker's gain for compromised device $\delta \in \mathcal{D}$
L_D^δ, L_A^δ	defender's / attacker's loss for compromised device $\delta \in \mathcal{D}$ (represented as negative values)
Variables	
$\boldsymbol{p}$	defender's strategy vector
$\boldsymbol{a}$	attacker's strategy vector
a_δ	attacker's action (i.e., attack or not) against device $\delta \in \mathcal{D}$
p_δ^m	probability of running attestation method $m \in \mathcal{M}$ on device $\delta \in \mathcal{D}$
Functions	
$P_\delta(\boldsymbol{p})$	conditional probability of defender detecting with strategy $\boldsymbol{p}$ that device $\delta \in \mathcal{D}$ is compromised (given that it is actually compromised)
$C_D^T(\boldsymbol{p})$	defender's total cost for strategy $\boldsymbol{p}$
$C_A^T(\boldsymbol{a})$	attacker's total cost for strategy $\boldsymbol{p}$
$U_D(\boldsymbol{p}, \boldsymbol{a})$	defender's expected utility for strategy profile $(\boldsymbol{p}, \boldsymbol{a})$
$U_A(\boldsymbol{p}, \boldsymbol{a})$	attacker's expected utility for strategy profile $(\boldsymbol{p}, \boldsymbol{a})$
$U_D^\delta(p_\delta^m, a_\delta)$	defender's expected utility from device $\delta \in \mathcal{D}$
$U_A^\delta(p_\delta^m, a_\delta)$	attacker's expected utility from device $\delta \in \mathcal{D}$
$\mathcal{F}_A(\boldsymbol{p})$	attacker's best response against defender strategy $\boldsymbol{p}$

Defender's Utility Different attestation methods can have different detection rates (i.e., different probability of detecting an attack when the method is run on a compromised device). For each attestation method $m \in \mathcal{M}$, we let μ^m denote the probability that method m detects that the device is compromised.

However, the defender can run multiple attestation methods on the same device, and any one of these may detect the compromise. Therefore, the probability $P_\delta(\boldsymbol{p})$ of detecting that device $\delta \in \mathcal{D}$ is compromised when the defender uses attestation strategy $\boldsymbol{p}$ is

$$P_\delta(\boldsymbol{p}) = 1 - \prod_{m \in \mathcal{M}} (1 - \mu^m \cdot p_\delta^m). \tag{1}$$

Each attestation method also has a computational cost, which the defender incurs for running the method on a device. For each attestation method $m \in \mathcal{M}$, we let C_D^m be the cost of running method m on a device. Then, the defender's expected total cost $C_D^T(\boldsymbol{p})$ for running attestation following strategy $\boldsymbol{p}$ is

$$C_D^T(\boldsymbol{p}) = \sum_{\delta \in \mathcal{D}} \sum_{m \in \mathcal{M}} C_D^m \cdot p_\delta^m. \tag{2}$$

Note that the expected total cost of attestation $C_D^T(\boldsymbol{p})$ depends on the probability of running attestation (higher the probability p_δ^m, higher the expected cost for device δ and method m).

Next, we let G_D^δ be the defender's gain when the attacker chooses to attack device $\delta \in \mathcal{D}$ and the defender detects that the device is compromised. On the other hand, let L_D^δ be the defender's loss when the attacker chooses to attack device δ and the defender does not detect that the device is compromised. Then, we can express the defender's expected utility $U_D(\boldsymbol{p}, \boldsymbol{a})$ when the defender uses attestation strategy $\boldsymbol{p}$ and the attacker uses attack strategy $\boldsymbol{a}$ as

$$U_D(\boldsymbol{p}, \boldsymbol{a}) = \sum_{\delta \in \mathcal{D}} \left[G_D^\delta \cdot P_\delta(\boldsymbol{p}) + L_D^\delta \cdot (1 - P_\delta(\boldsymbol{p})) \right] \cdot a_\delta - C_D^T(\boldsymbol{p}). \tag{3}$$

Attacker's Utility Let $C_A^\mathcal{E}$ be the cost of developing an exploit for device class $\mathcal{E}$, and let C_A^δ be the cost of attacking a particular device $\delta \in \mathcal{D}$. For any attack strategy $\boldsymbol{a}$, the set of device classes that the adversary attacks can be expressed as $\{\mathcal{E} \mid \exists\, \delta \in \mathcal{E}\ (a_\delta = 1)\}$. Then, we can express the adversary's total cost $C_A^T(\boldsymbol{a})$ for attack strategy $\boldsymbol{a}$ as

$$C_A^T(\boldsymbol{a}) = \sum_{\mathcal{E}} \left(C_A^\mathcal{E} \cdot 1_{\{\exists \delta \in \mathcal{E}(a_\delta = 1)\}} + \sum_{\delta \in \mathcal{E}} C_A^\delta \cdot a_\delta \right). \tag{4}$$

Note that the attacker incurs cost for both developing an exploit for each class that it targets as well as for each individual device.

Similar to the defender, we let attacker's gain and loss for attacking a device $\delta \in \mathcal{D}$ be G_A^δ and L_A^δ when the compromise is not detected and detected, respectively. Then, we can express the adversary's expected utility $U_A(\boldsymbol{p}, \boldsymbol{a})$ when the defender uses attestation strategy $\boldsymbol{p}$ and the attacker uses attack strategy $\boldsymbol{a}$ as

$$U_A(\boldsymbol{p}, \boldsymbol{a}) = \sum_{\delta \in \mathcal{D}} \left[L_A^\delta \cdot P_\delta(\boldsymbol{p}) + G_A^\delta \cdot (1 - P_\delta(\boldsymbol{p})) \right] \cdot a_\delta - C_A^T(\boldsymbol{a}). \tag{5}$$

For the sake of simplicity, we assume that with respect to gains and losses from compromises, the players' utilities are zero sum, that is, $G_D^\delta = -L_A^\delta$ and $L_D^\delta = -G_A^\delta$. Note that the game is not zero sum due to the players' asymmetric costs $C_D^T(\boldsymbol{p})$ and $C_A^T(\boldsymbol{a})$.

4.4 Solution Concept

We assume that both the defender and attacker aim to maximize their expected utilities. To formulate the optimal attestation strategy for the defender, we first define the attacker's best-response strategy.

In response to a defender's strategy $\boldsymbol{p}$, the attacker always chooses an attack strategy $\boldsymbol{a}$ that maximizes the attacker's expected utility $U_A(\boldsymbol{p}, \boldsymbol{a})$. Therefore, we can define the attacker's best response as follows.

Definition 1 (Attacker's best response). *Against a defender strategy $\boldsymbol{p}$, the attacker's* best-response strategy $\mathcal{F}_A(\boldsymbol{p})$ *is*

$$\mathcal{F}_A(\boldsymbol{p}) = \operatorname{argmax}_{\boldsymbol{a}} U_A(\boldsymbol{p}, \boldsymbol{a}). \tag{6}$$

Note that the best response is not necessarily unique (i.e., $\mathcal{F}_A$ may be a set of more than one strategies). Hence, as is usual in the literature, we will assume tie-breaking in favor of the defender to formulate the optimal attestation strategy.

Since the defender's objective is to choose an attestation strategy $\boldsymbol{p}$ that maximizes its expected utility $U_D(\boldsymbol{p}, \boldsymbol{a})$ anticipating that the attacker will choose a best-response strategy from $\mathcal{F}(\boldsymbol{p})$, we can define the defender's optimal strategy as follows.

Definition 2 (Defender's optimal strategy). *The defender's* optimal attestation strategy $\boldsymbol{p}^*$ *is*

$$\boldsymbol{p}^* = \operatorname{argmax}_{\boldsymbol{p},\, \boldsymbol{a} \in \mathcal{F}_A(\boldsymbol{p})} U_D(\boldsymbol{p}, \boldsymbol{a}) \tag{7}$$

5 Analysis of Optimal Attestation Strategies

Here, we present analytical results on our game-theoretic model, characterizing the defender's optimal strategy in important special cases. For ease of exposition, we present these special cases in increasing generality. Due to lack of space, we omit proofs and details of the analysis in this document. The proofs and details are available in the extended online version [12].

5.1 Case 1: Single Device and Single Attestation Method

First, we assume that there exists only one device δ and one attestation method m.

Attacker's Best-Response Strategy Whether the attacker's best response is to attack or not depends on the defender's strategy p_δ^m. Further, it is easy to see that if attacking is a best response for some p_δ^m, then it must also be a best response for any $\hat{p}_\delta^m < p_\delta^m$. Therefore, there must exist a threshold value τ_δ of the defender's probability p_δ^m that determines the attacker's best response.

Lemma 1. *The attacker's best-response strategy* $\mathcal{F}(\boldsymbol{p})$ *is*

$$\mathcal{F}(\boldsymbol{p}) = \begin{cases} \{1\} & \text{if } p_\delta^m < \tau_\delta \\ \{0,1\} & \text{if } p_\delta^m = \tau_\delta \\ \{0\} & \text{otherwise,} \end{cases} \tag{8}$$

where

$$\tau_\delta = \frac{1}{\mu^m} \cdot \frac{C_A^{\mathcal{E}} + C_A^\delta - G_A^\delta}{L_A^\delta - G_A^\delta}. \tag{9}$$

In other words, it is a best response for the attacker to attack if the defender's attestation probability p_δ^m is lower than the threshold τ_δ; and it is a best response not to attack if the probability p_δ^m is higher than the threshold τ_δ.

Defender's Optimal Strategy The defender may pursue one of two approaches for maximizing its own expected utility: selecting an attestation probability that is high enough to deter the attacker from attacking (i.e., to eliminate losses by ensuring that not attacking is a best response for the attacker); or selecting an attestation probability that strikes a balance between risk and cost, accepting that the adversary might attack.

First, from Equations (2) and (8), it is clear that the lowest-cost strategy for deterring the attacker is $p_\delta^m = \tau_\delta$. Second, if the defender does not deter the attacker, then it must choose a probability p_δ^m from the range $[0, \tau_\delta]$ that maximizes $U_D(p_\delta^m, 1)$. Then, it follows from Equation (3) that the optimal probability is either $p_\delta^m = 0$ or τ_δ, depending on the constants μ^m, C_D^m, G_D^δ, and L_D^δ.

Proposition 1. *The defender's optimal attestation strategy* ${p^*}_\delta^m$ *is*

$${p^*}_\delta^m = \begin{cases} 0 & \text{if } C_D^m \geq (G_D^\delta - L_D^\delta) \cdot \mu^m \text{ and } \tau_\delta \geq \frac{L_D^\delta}{-C_D^m} \\ \tau_\delta & \text{otherwise.} \end{cases} \tag{10}$$

Note that the first case corresponds to when deterrence is not necessarily better than non-deterrence (first condition), and for non-deterrence strategies, minimizing risks over costs is not better (second condition).

5.2 Case 2: Multiple Devices and Single Device Class

Next, we generalize our analysis by allowing multiple devices $\mathcal{D}$, but assuming a single device class $\mathcal{E} = \mathcal{D}$ and single attestation method m.

Attacker's Best-Response Strategy First, the attacker needs to decide whether it will attack at all: if the attacker does not attack at all, it attains $U_A(\boldsymbol{p}, \mathbf{0}) = 0$ utility; if the attacker does attack some devices, it incurs the cost $C_A^{\mathcal{E}}$ of attacking the class once, and it will need to make decisions for each individual device $\delta \in \mathcal{D}$ without considering this cost $C_A^{\mathcal{E}}$. The latter is very similar to *Case 1* since for each individual device δ, the decision must be based on a threshold value τ_δ of the attestation probability p_δ^m; however, this threshold must now ignore $C_A^{\mathcal{E}}$.

Lemma 2. *The attacker's best-response strategy* $\mathcal{F}(\boldsymbol{p})$ *is*

$$\mathcal{F}(\boldsymbol{p}) = \begin{cases} \{\boldsymbol{a}^*\} & \text{if } U_A(\boldsymbol{p}, \boldsymbol{a}^*) > 0 \\ \{\boldsymbol{a}^*, \boldsymbol{0}\} & \text{if } U_A(\boldsymbol{p}, \boldsymbol{a}^*) = 0 \\ \{\boldsymbol{0}\} & \text{otherwise,} \end{cases} \tag{11}$$

where

$$a^*_\delta = \begin{cases} 1 & \text{if } p^m_\delta < \overline{\tau}_\delta \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

and

$$\overline{\tau}_\delta = \frac{1}{\mu^m} \cdot \frac{C^\delta_A - G^\delta_A}{L^\delta_A - G^\delta_A}. \tag{13}$$

Note that strategy $\boldsymbol{a}^*$ is a utility-maximizing strategy for the attacker assuming that it has already paid the cost $C^{\mathcal{E}}_A$ for attacking the class. Hence, the decision between attacking (in which case $\boldsymbol{a}^*$ is optimal) and not attacking at all ($\boldsymbol{a} = \boldsymbol{0}$) can is based on the utility $U_A(\boldsymbol{p}, \boldsymbol{a}^*)$ obtained from strategy $\boldsymbol{a}^*$ and the utility $U_A(\boldsymbol{p}, \boldsymbol{0}) = 0$ obtained from not attacking at all.

Defender's Optimal Strategy Again, the defender must choose between deterrence and acceptance (i.e., deterring the adversary from attacking or accepting that the adversary might attack). However, in contrast to *Case 1*, the defender now has the choice between completely deterring the adversary from attacking (i.e., adversary is not willing to incur cost $C^{\mathcal{E}}_A$ and hence attacks no devices at all) and deterring the adversary only from attacking some devices (i.e., adversary incurs cost $C^{\mathcal{E}}_A$ and attacks some devices, but it is deterred from attacking other devices).

Proposition 2. *The defender's optimal attestation strategy* $\boldsymbol{p}^*$ *is*

$$\boldsymbol{p}^* = \begin{cases} \{\boldsymbol{p}^{ND}\} & \text{if } U_D\left(\boldsymbol{p}^{ND}, \boldsymbol{a}^*\right) > U_D\left(\boldsymbol{p}^D, \boldsymbol{0}\right) \\ \{\boldsymbol{p}^{ND}, \boldsymbol{p}^D\} & \text{if } U_D\left(\boldsymbol{p}^{ND}, \boldsymbol{a}^*\right) = U_D\left(\boldsymbol{p}^D, \boldsymbol{0}\right) \\ \{\boldsymbol{p}^D\} & \text{otherwise,} \end{cases} \tag{14}$$

where

$$\left(p^{ND}\right)^m_\delta = \begin{cases} 0 & \text{if } C^m_D \geq (G^\delta_D - L^\delta_D) \cdot \mu^m \text{ and } \overline{\tau}_\delta \geq \frac{L^\delta_D}{-C^m_D} \\ \overline{\tau}_\delta & \text{otherwise,} \end{cases} \tag{15}$$

$\boldsymbol{a}^*$ *is as defined in Equation* (12) *with* $\boldsymbol{p} = \boldsymbol{p}^{ND}$, *and*

$$\boldsymbol{p}^D = \operatorname{argmin}_{\left\{\boldsymbol{p}:\, U_A(\boldsymbol{p},\boldsymbol{1}) \leq 0 \,\wedge\, \forall\delta\left(p^m_\delta \in [0,\tau_\delta]\right)\right\}} \sum_{\delta \in \mathcal{D}} C^m_D \cdot p^m_\delta. \tag{16}$$

Note that $\boldsymbol{p}^{\mathrm{ND}}$ is the optimal attestation strategy if the defender does not completely deter the adversary from attacking, calculated similarly to *Case 1*; $\boldsymbol{p}^{\mathrm{D}}$ is the optimal attestation strategy if the defender completely deters the adversary from attacking, which may be computed by solving a simple linear optimization (Equation 16).

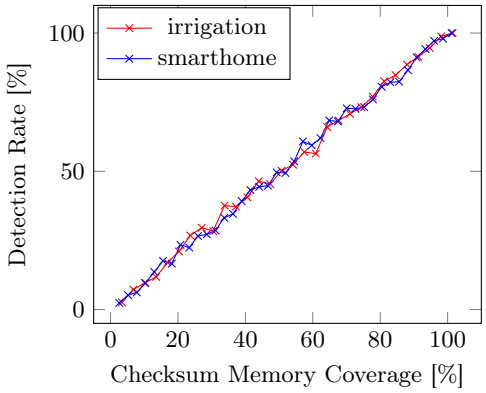


Fig. 2: Detection rate of pseudo-random memory checksum as a function of memory coverage.

5.3 Case 3: Multiple Devices and Multiple Device Classes

Next, we consider multiple device classes $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$. We generalize our previous results by observing that both the attacker's and defender's decisions for each class of devices are independent of other classes.

Lemma 3. *For each device class $\mathcal{E}_i$, let $\boldsymbol{a}_i$ be a best response as given by Lemma 2. Then, $\langle \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n \rangle$ is a best-response attack strategy.*

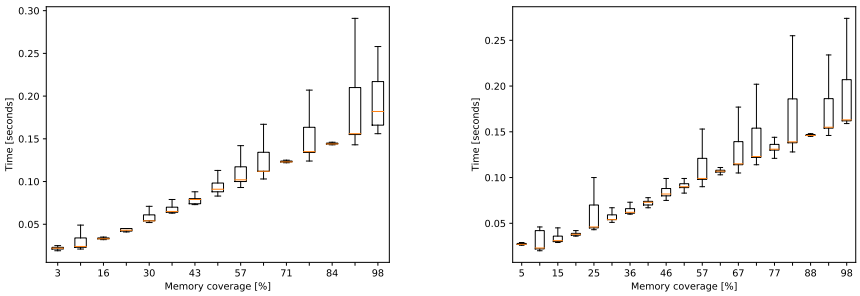
Proposition 3. *For each device class $\mathcal{E}_i$, let $\boldsymbol{p}_i^*$ be an optimal attestation strategy as given by Proposition 2. Then, $\langle \boldsymbol{p}_1^*, \boldsymbol{p}_2^*, \ldots, \boldsymbol{p}_n^* \rangle$ is an optimal attestation strategy.*

6 Numerical Results

Here, we present experimental results from our testbed, which confirm our modeling assumptions, as well as numerical results on our game-theoretic model.

6.1 Experimental Results from the Remote Attestation Testbed

We consider an experimental setup with two test applications, an irrigation and a smarthome application. We implement sequential and pseudo-random memory checksum as exemplary software-based remote attestation methods. Software-based attestation incurs various costs; in this section, we study checksum-based remote attestation in terms of memory and computational overhead. We also evaluate checksum-based attestation in terms of detection rate.



(a) irrigation (b) smarthome

Fig. 3: Running time of checksum calculation as a function of memory coverage.

Detection Rate of Pseudo-random Memory Checksum In this experiment, we study the efficacy of pseudo-random memory checksum in terms of detecting changes to the code of a running application. We use a block-based pseudo-random technique, where each block is 500 bytes. We start our experiment with checking 200 blocks, which are selected pseudo-randomly based on a seed value. Then, we increase the number of blocks by 200 in iterations to measure the impact of increasing memory coverage. In each iteration, we run the pseudo-random memory checksum 500 times, using a different random seed each time, to reliably measure the detection rate for a certain number of blocks.

Figure 2 shows the detection rate of pseudo-random memory checksum as a function of the fraction of memory covered by the checksum calculation, for our two test applications. Note that we express the fraction of memory covered as a percentage. Specifically, we calculate the ratio as (number of blocks$\times$block size$\times$ 100)/total memory size of the program. We find that detection rate increases roughly proportionally with memory coverage, ranging from 0% to 100%, which supports our modeling choices.

Running Time of Pseudo-Random Memory Checksum Next, we study the running time of calculating pseudo-random memory checksum with memory coverage ranging from 3% to 98%. For each memory-coverage level, we run the checksum calculation 500 times to obtain reliable running-time measurements. Figure 3 shows the distribution of running time for various memory-coverage level for our two test applications. We find that similar to detection rate, the average of running time also increases proportionally with memory coverage, which supports our modeling choices.

6.2 Evaluation of Game-Theoretic Model and Optimal Strategies

To evaluate our model and optimal strategies, we consider an example environment consisting of $|\mathcal{D}| = 50$ IoT devices from 5 different classes (10 devices in

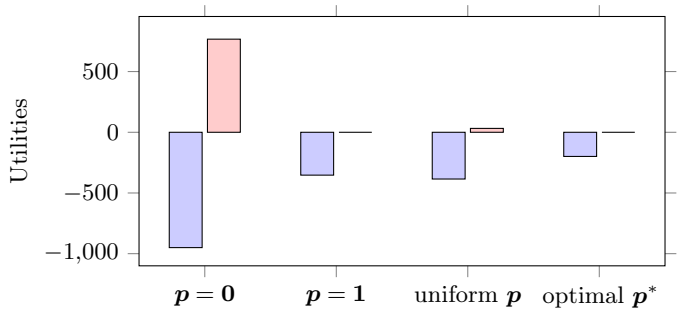
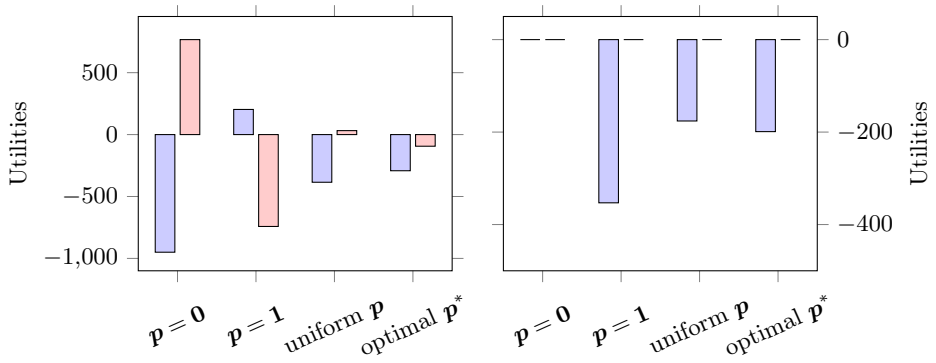


Fig. 4: Comparison between optimal and naïve defender strategies based on the defender's utility (blue ▯) and the attacker's utility (red ▯), assuming that the attacker chooses its best response.

each class $\mathcal{E}_i$), and for simplicity, we consider a single attestation method m implemented in these devices. For each device δ, we choose both the defender's and the attacker's gain values G_D^δ and G_A^δ uniformly at random from $[20, 40]$. We assume that the game is zero-sum with respect to gains and losses; that is, we let the players' losses (L_D^δ, L_A^δ) be $G_D^\delta = -L_A^\delta$ and $L_D^\delta = -G_A^\delta$. Finally, we choose the detection probability of the attestation method μ uniformly at randomly from $[0.5, 0.9]$, the attestation cost C_D from $[0, 10]$, the exploit development cost $C_A^{\mathcal{E}}$ from $[15, 40]$, and the device attack costs C_A^δ from $[1, 3]$ for each device δ.

Comparison to Naïve Baselines We compare the defender's optimal attestation strategy $\boldsymbol{p}^*$ to three naïve baseline strategies: $\boldsymbol{p} = \mathbf{0}$, $\boldsymbol{p} = \mathbf{1}$, and an optimal uniform $\boldsymbol{p}$ (i.e., same probability p_δ^m for all devices δ, but this probability is chosen to maximize the defender's utility given that the adversary always chooses its best response). Figure 4 shows the players' utilities for the optimal and naïve defender strategies, assuming that the adversary chooses its best response in each case. We see that the optimal strategy outperforms the naïve baselines in terms of the defender's utility. Specifically, it outperforms $\boldsymbol{p} = \mathbf{0}$ and optimal uniform $\boldsymbol{p}$ by deterring the adversary from attacking, which these naïve baselines fail to achieve; and it outperforms $\boldsymbol{p} = \mathbf{1}$ by achieving deterrence at a lower cost.

Detailed Comparison to Naïve Baselines Figure 5 provides a more detailed comparison between the optimal attestation strategy $\boldsymbol{p}^*$ and the three naïve baselines. In contrast to Figure 4, this figure shows utilities both in the case when the adversary decides to attack (Figure 5a) and in the case when it decides to not attack at all (Figure 5b). In Figure 5a, we see that the adversary can obtain a positive utility from attacking against $\boldsymbol{p} = \mathbf{0}$ and the optimal uniform $\boldsymbol{p}$. Therefore, these strategies do not deter the adversary from attacking. In contrast, the adversary's utility is negative against both $\boldsymbol{p} = \mathbf{1}$ and the optimal strategy $\boldsymbol{p}^*$. In Figure 5b, we see that the defender incurs higher computational cost with $\boldsymbol{p} = \mathbf{1}$ than with the optimal strategy $\boldsymbol{p}^*$, making the latter the better choice.



(a) Attacker chooses to attack (b) Attacker chooses not to attack

Fig. 5: Detailed comparison between optimal and naïve defender strategies based on the defender's utility (blue ▯) and the attacker's utility (red ▯).

Comparison of Strategy Profiles To better understand how the optimal attestation strategy outperforms the other strategies, we now take a closer look at the players' utilities in specific strategy profiles. For the defender, we consider two strategies: optimal strategy given that the defender does not completely deter the attacker ($\boldsymbol{p}^{\mathrm{ND}}$, see Proposition 2) and optimal strategy that completely deters the attacker ($\boldsymbol{p}^{\mathrm{D}}$, which is the optimal attestation strategy in this problem instance). For the adversary, we also consider two strategies: attacking every device ($\boldsymbol{a} = \mathbf{1}$, which is a best response against $\boldsymbol{p}^{\mathrm{ND}}$ in this problem instance) and not attacking at all ($\boldsymbol{a} = \mathbf{0}$, which is always a best response against $\boldsymbol{p}^{\mathrm{D}}$).

Figure 6 shows the players' utilities in the four strategy profiles formed by the above strategies. We observe that the defender's utility is highest when it does not completely deter the adversary from attacking and the adversary does not attack at all ($\boldsymbol{p}^{\mathrm{ND}}$ vs. $\boldsymbol{a} = \mathbf{0}$) since the defender incurs minimal computational cost and suffers no security losses in this case. However, this is not an equilibrium since the adversary can attain higher utility by attacking every device (see $\boldsymbol{p}^{\mathrm{ND}}$ vs. $\boldsymbol{a} = \mathbf{1}$), which results in the best utility for the adversary and worst for the defender. To avoid such catastrophic losses, the defender can use the strategy of complete deterrence, in which case the adversary will be indifferent between attacking and not attacking ($\boldsymbol{p}^{\mathrm{D}}$ vs. $\boldsymbol{a} = \mathbf{0}$ and $\boldsymbol{a} = \mathbf{1}$). Note that since the defender's utility is higher if the adversary does not attack, the defender can opt to tip the balance in favor of not attacking through an infinitesimal change.

7 Related Work

In this section, first we discuss the pros and cons of different IoT testbeds in the existing literature. Then we discuss existing works related to hardware- or software-based IoT remote attestation. Finally, we present a few SSG works and how our approach is different from theirs.

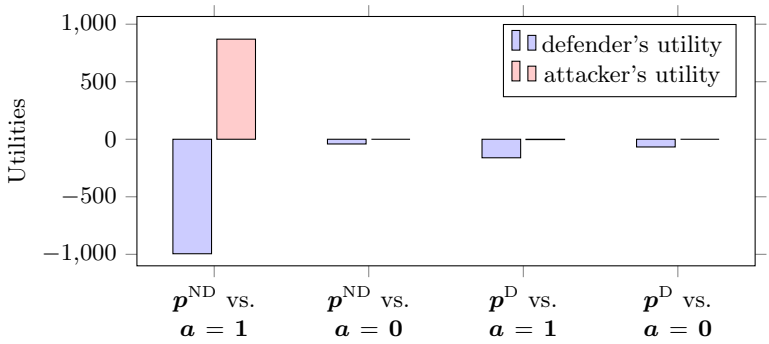


Fig. 6: Players' utilities in various strategy profiles: not deter vs. attack ($\boldsymbol{p}^{\mathrm{ND}}$ vs. $\boldsymbol{a} = \mathbf{1}$), not deter vs. not attack ($\boldsymbol{p}^{\mathrm{ND}}$ vs. $\boldsymbol{a} = \mathbf{0}$), deter vs. attack ($\boldsymbol{p}^{\mathrm{D}}$ vs. $\boldsymbol{a} = \mathbf{1}$), deter vs. not attack ($\boldsymbol{p}^{\mathrm{D}}$ vs. $\boldsymbol{a} = \mathbf{0}$).

7.1 IoT Security Testbeds

General application- or hardware-oriented IoT testbeds are widely used for research works. The primary concern for these testbeds are to find ideal configurations or set up in different types of environments [4,2]. Several IoT security testbeds are available to test different security and reliability issues. For example, Siboni et al. proposed an IoT security testbed framework, arguing that an ideal testbed should ensure reliability, anti-forensics, and adaptivity [15]. In another work, Arseni et al. developed a heterogeneous IoT testbed named *Pass-IoT*, consisting of three different architectures (MCU, SDSoC, traditional CPU) that can test, optimize, and develop lightweight cryptographic algorithms [3].

Nowadays, the number of IoT applications is rising, and so are associated security concerns for these applications. Therefore, Tekeoglu et al. developed a security testbed that can perform privacy analysis of IoT devices, including HDMI sticks, IP cameras, smartwatches, and drones [18]. The testbed enables identifying insecure protocol versions, authentication issues, and privacy violations.

We find the existing IoT security testbeds offering general security concerns related to cryptographic development, secure protocol implementations, and data privacy issues. Our remote attestation testbed offers testing application vulnerabilities, developing associated exploits, and evaluating mitigation measures through software-oriented remote attestation. Table 2 presents a comparative analysis between our work and the existing IoT security testbeds.

7.2 Remote Attestation

Checksum-based remote attestation has been widely used to secure IoT devices for a long time. Earlier, Seshadri et al. proposed a cell-based pseudo-random traversal approach in their software-based attestation scheme entitled *SWATT* [14]. The authors developed the 8-bit micro-controller architecture scheme to generate random addresses to checksum using an RC4 stream cipher.

Table 2: IoT Security Testbeds

Features \ Works	Arseni et al., 2016	Tekeoglu et al., 2016	Our Work
Lightweight Encryption Algorithms Development	✓	X	X
Vulnerability Scans	X	✓	✓
Authentication, Privacy Violations	X	✓	X
Exploitation Development and Analysis	X	X	✓
Mitigation Measures Testing	X	X	✓
Remote Attestation Experiments	X	X	✓

Yang et al. proposed a distributed software-based attestation scheme for WSN to verify the integrity of code in a distributed fashion [23]. The works led to later works in terms of cell- and block-based pseudo-random checksum, respectively.

A few recent works include hardware-assisted remote runtime attestation [8] that addresses runtime attack detection, a low-cost checksum-based remote memory attestation for smart grid [22], lightweight remote attestation in distributed wireless sensor networks where all nodes validate each other's data [9], and so on. Survey papers on attestation, for example, Steiner et al. presented a more comprehensive overview on checksum-based attestation [17].

7.3 Stackelberg Security Games

SSGs have been successfully applied to security problems in resource-limited domains such as airport security, biodiversity protection, randomized inspections, border patrols, cyber security, and so on [19,7,5]. To the best of our knowledge, our work is the first to apply game theory in the area of remote attestation.

While there is no prior work within the intersection of game theory and remote attestation, a number of research efforts have applied SSGs to other detection problems that resemble ours. For example, Wahab et al. developed a Bayesian Stackelberg game that helps the defender to determine optimal detection load distribution strategy among virtual machines within a cloud environment [20]. As another example, Chen et al. develops an SSG model that detects adversarial outbreak in an IoT environment through determining strategic dynamic scheduling of intrusion detection systems [6].

8 Conclusion and Future Work

IoT device exploitation has been a significant issue lately, and organizations are investing significant resources and effort into managing these security risks. An important approach for mitigating this threat is remote attestation, which enables the defender to remotely verify the integrity of devices and their software. In this work, we developed a testbed that offers research opportunities to explore and analyze IoT vulnerabilities and exploitation and to conduct experiments with varous remote attestation methods.

So far, we have developed attack strategies mostly for when kernel and compiler-based security measures are disabled. In future work, we plan to include exploitation with security features enabled in the resource-limited IoT environment. Additionally, in this paper, we evaluated software-based attestation methods (sequential and random memory-based checksum). We intend to include some other variants of attestation methods (e.g., hybrid checksum) in the testbed and to conduct experiments with control-flow integrity.

Further, we have showed how to optimize remote-attestation strategies by formulating and studying a Stackelberg security game model. Our analytical results provide algorithmic solutions for finding optimal attestation strategies in a variety of settings. These results can provide guidance to practitioners on how to protect IoT devices using remote attestation in resource-limited environments.

In this work we discussed optimal strategies for one attestation method ($|\mathcal{M}| = 1$). In future, we plan to provide analytical solutions for more general cases of multiple devices ($|\mathcal{D}| > 1$), multiple classes ($|\mathcal{E}| > 1$), and multiple attestation methods ($|\mathcal{M}| > 1$) as well. Additionally, we intend to refine our model and find optimal strategies in a more complex environment using machine learning algorithms (e.g., reinforcement learning).

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. CNS-1850510, IIS-1905558, and ECCS-2020289 and by the Army Research Office under Grant No. W911NF1910241 and W911NF1810208.

References

1. Abera, T., Asokan, N., Davi, L., Koushanfar, F., Paverd, A., Sadeghi, A.R., Tsudik, G.: Things, trouble, trust: on building trust in IoT systems. In: Proceedings of the 53rd Annual Design Automation Conference. pp. 1–6 (2016)
2. Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., et al.: FIT IoT-LAB: A large scale open experimental IoT testbed. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). pp. 459–464. IEEE (2015)
3. Arseni, Ş.C., Miţoi, M., Vulpe, A.: Pass-IoT: A platform for studying security, privacy and trust in IoT. In: 2016 International Conference on Communications (COMM). pp. 261–266. IEEE (2016)
4. Belli, L., Cirani, S., Davoli, L., Gorrieri, A., Mancin, M., Picone, M., Ferrari, G.: Design and deployment of an IoT application-oriented testbed. Computer **48**(9), 32–40 (2015)
5. Bucarey, V., Casorrán, C., Figueroa, Ó., Rosas, K., Navarrete, H., Ordóñez, F.: Building real Stackelberg security games for border patrols. In: International Conference on Decision and Game Theory for Security. pp. 193–212. Springer (2017)
6. Chen, L., Wang, Z., Li, F., Guo, Y., Geng, K.: A stackelberg security game for adversarial outbreak detection in the internet of things. Sensors **20**(3), 804 (2020)
7. Gan, J., Elkind, E., Wooldridge, M.: Stackelberg security games with multiple uncoordinated defenders. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. pp. 703–711 (2018)

8. Geden, M., Rasmussen, K.: Hardware-assisted remote runtime attestation for critical embedded systems. In: 2019 17th International Conference on Privacy, Security and Trust (PST). pp. 1–10. IEEE (2019)
9. Kiyomoto, S., Miyake, Y.: Lightweight attestation scheme for wireless sensor network. International Journal of Security and Its Applications **8**(2), 25–40 (2014)
10. Nunes, I.D.O., Eldefrawy, K., Rattanavipanon, N., Steiner, M., Tsudik, G.: Vrased: A verified hardware/software co-design for remote attestation. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 1429–1446 (2019)
11. Parikh, V., Mateti, P.: Aslr and rop attack mitigations for arm-based android devices. In: International Symposium on Security in Computing and Communication. pp. 350–363. Springer (2017)
12. Roy, S., Kadir, S.U., Vorobeychik, Y., Laszka, A.: Strategic remote attestation: Testbed for Internet-of-Things devices and Stackelberg security game for optimal strategies. arXiv preprint arXiv:2109.07724 (2021)
13. Seshadri, A., Luk, M., Shi, E., Perrig, A., Van Doorn, L., Khosla, P.: Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms. In: Proceedings of ACM Symposium on Operating Systems Principles (SOSP). vol. 173, pp. 10–1145 (2005)
14. Seshadri, A., Perrig, A., Van Doorn, L., Khosla, P.: Swatt: Software-based attestation for embedded devices. In: IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004. pp. 272–282. IEEE (2004)
15. Siboni, S., Sachidananda, V., Meidan, Y., Bohadana, M., Mathov, Y., Bhairav, S., Shabtai, A., Elovici, Y.: Security testbed for internet-of-things devices. IEEE Transactions on Reliability **68**(1), 23–44 (2019)
16. Sinha, A., Fang, F., An, B., Kiekintveld, C., Tambe, M.: Stackelberg security games: Looking beyond a decade of success. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. IJCAI (2018)
17. Steiner, R.V., Lupu, E.: Attestation in wireless sensor networks: A survey. ACM Computing Surveys (CSUR) **49**(3), 1–31 (2016)
18. Tekeoglu, A., Tosun, A.Ş.: A testbed for security and privacy analysis of IoT devices. In: 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). pp. 343–348. IEEE (2016)
19. Trejo, K.K., Clempner, J.B., Poznyak, A.S.: Adapting strategies to dynamic environments in controllable Stackelberg security games. In: 2016 IEEE 55th Conference on Decision and Control (CDC). pp. 5484–5489. IEEE (2016)
20. Wahab, O.A., Bentahar, J., Otrok, H., Mourad, A.: Resource-aware detection and defense system against multi-type attacks in the cloud: Repeated bayesian stackelberg game. IEEE Transactions on Dependable and Secure Computing **18**(2), 605–622 (2019)
21. Xu, B., Wang, W., Hao, Q., Zhang, Z., Du, P., Xia, T., Li, H., Wang, X.: A security design for the detecting of buffer overflow attacks in IoT device. IEEE Access **6**, 72862–72869 (2018)
22. Yang, X., He, X., Yu, W., Lin, J., Li, R., Yang, Q., Song, H.: Towards a low-cost remote memory attestation for the smart grid. Sensors **15**(8), 20799–20824 (2015)
23. Yang, Y., Wang, X., Zhu, S., Cao, G.: Distributed software-based attestation for node compromise detection in sensor networks. In: 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007). pp. 219–230. IEEE (2007)
24. Yin, Z., Korzhyk, D., Kiekintveld, C., Conitzer, V., Tambe, M.: Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1. pp. 1139–1146 (2010)