

Ubiquitous Sensing for Security in Smart Homes



Simon Birnbach
Worcester College
University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2021

Acknowledgements

Institutional

With sincere gratitude I thank the EPSRC for funding the tuition fees of my DPhil programme, the Department of Computer Science for the studentship covering my living costs, Worcester College for providing funds that allowed me to present my work at international conferences, and MasterCard for their generous financial support in building up our IoT lab, which enabled me to pursue this work.

Personal

I extend my deepest thanks to my supervisor Ivan Martinovic, who believed in me and supported me even before I embarked on this DPhil. Your supervision and guidance has allowed me to grow both as a researcher and as a person.

I also owe a debt of gratitude to Andrew Markham, Kasper Rasmussen and Asokan, who assessed my work at various points and whose valuable feedback has helped shape this thesis.

I am particularly indebted to Simon Eberz and Richard Baker, who have proved to be not only amazing co-authors, but also dear friends.

I feel grateful for the time I got to spend with everyone who has been part of my research group over the past few years, particularly those with whom I shared an office. You have helped me get through the many long days at the department.

I want to thank my friends, both in Oxford and across the world, who have provided the much needed counterbalance to the stress at work.

Last but certainly not least, I want to say thanks to those dearest to me. To my parents, for their love and encouragement throughout all my life. I would not be where I am today without you. To my sister, for always being there for me when I need her. To Gabi, for her unwavering support and for helping me formulate my thoughts when I was stuck.

Abstract

In recent years, smart home systems have become increasingly popular. With this increased popularity has come a newfound prevalence of IoT devices with varied sensor functionality in peoples' homes. These types of IoT devices are widely considered to be insecure and many problem-specific solutions have been proposed. However, we argue that the widespread presence of diverse sensors in and of itself can—and should—be used to strengthen the security of smart home environments.

In this thesis, we show that the ubiquity of sensors in modern smart homes can improve the overall security of smart home systems, and that it can enable new security functions. To this end, we introduce two security systems based on ubiquitous sensors already available in modern smart homes. The first is a drone detection system that relies only on radio signal strength information that can be provided by most wireless consumer devices. The second is an event verification system for event-driven smart homes that uses physical event signatures from a multitude of sensors to verify the occurrence of physical events in smart home environments. We then analyze how such systems are affected by two advanced types of attackers: an opportunistic attacker who can wait for and predict attack opportunities, and an attacker who can compromise sensor data to create these opportunities themselves. Finally, we demonstrate that contextual information widely available in smart homes can be employed for context-aware sensor selection due to improved situational awareness.

Extrapolating from the insights gained within this thesis, we argue that ubiquitous sensing has significant potential to provide stronger security properties in smart home systems, as it allows for the combination of fine-grained sensor data with high-level event information. Crucially, this added security can be achieved with existing smart home devices—without impacting their original functions.

Contents

List of Figures	xi
List of Abbreviations	xiii
1 Introduction	1
1.1 Motivation	2
1.2 Contributions	3
1.3 Outline	5
2 Related Work	7
2.1 Security of Smart Home Systems	8
2.2 Compromised Devices	9
2.3 Mitigation of Compromised Devices	11
2.4 Context-based Pairing in Smart Home Systems	12
2.5 Attacks on Sensors	13
2.6 Attack-resilient Sensor Fusion	14
3 Background	17
3.1 Drone Detection and Countermeasures	18
3.1.1 Drone Detection	18
3.1.2 Countermeasures	22
3.2 Event Detection and Verification	22
3.2.1 Event-driven Smart Home Systems	23
3.2.2 Event Detection in Sensor Networks	25
3.2.3 Activity Recognition in Home Environments	27
3.2.4 Event Detection for Security	30
3.3 Metrics	31
3.4 Research Questions	33
4 Real-world Drone Detection Using Smart Home Devices	37
4.1 Introduction	38
4.2 Background	42
4.3 Adversary Model	44

4.4	System Model	46
4.4.1	Pre-processing	47
4.4.2	Statistical Tests	47
4.4.3	Attack Analysis	49
4.5	Standard Deviation-based Method	49
4.5.1	Attack Analysis	51
4.5.2	Detection Range	53
4.5.3	Approach Patterns	53
4.5.4	Detection Algorithm	55
4.6	Preliminary Experiment	58
4.7	Preliminary Evaluation	60
4.7.1	Straight Approach	61
4.7.2	Zig-zag Approach	65
4.7.3	Back-and-forth Approach	67
4.7.4	NLOS Approach Over the Roof	67
4.7.5	NLOS Approach Around the House	69
4.7.6	Detection Ranges	69
4.8	Discussion of Preliminary Results	72
4.8.1	Results	72
4.8.2	Observations	73
4.8.3	Deployment Considerations and Mobile Transmitters	73
4.9	Experiment with Mobile Transmitters and Interfering Traffic	74
4.9.1	Non-drone Approaches	75
4.9.2	Static Transmitters	75
4.9.3	Drone Approaches	76
4.10	Evaluation of the Standard Deviation-based Method	77
4.11	Kurtosis-based Method	79
4.11.1	Long-term Test	80
4.11.2	Short-term Test	81
4.11.3	Message Rate	82
4.11.4	Algorithm	82
4.12	Evaluation of the Kurtosis-based Method	82
4.12.1	Detection Performance	83
4.12.2	Spatial Dependence	85
4.12.3	Multiple Receivers	86
4.12.4	Message Rate	86
4.12.5	Detection Distance	89
4.12.6	Comparison With Other Systems	90
4.13	Limitations	92
4.14	Conclusion	94

5	Smart Home Event Verification in the Presence of Strong Attackers	97
5.1	Introduction	99
5.2	Background	103
5.2.1	Smart Home Systems	103
5.3	System Model	104
5.4	Threat Model	107
5.5	Experimental Design	109
5.5.1	Overview	109
5.5.2	Data Collection	112
5.5.3	Ethical Concerns	114
5.6	Methods	114
5.6.1	Data Labeling	114
5.6.2	Dataset Division	116
5.6.3	Sensor Window Selection	117
5.6.4	Feature Extraction	117
5.6.5	Classification and Metrics	119
5.6.6	Attacker Implementation	121
5.7	Results	123
5.7.1	Feature Distinctiveness	123
5.7.2	Spoofing Attack Detection	126
5.7.3	Masking Attack Detection	128
5.7.4	Opportunistic Attacker	129
5.7.5	Sensor-compromise Attacker	132
5.8	Discussion	137
5.9	Conclusion	139
6	Context-aware Sensor Selection in Smart Environments	141
6.1	Introduction	142
6.2	Context-aware System Model Extension	144
6.3	Threat Model	146
6.4	Methods	147
6.4.1	Pre-processing	147
6.4.2	Sensor Fusion	147
6.4.3	Context-aware Attacker Behavior	151
6.5	Evaluation	154
6.5.1	Impact of Context-awareness	154
6.5.2	Sensor-compromise Attacker	158
6.5.3	Context Selection and Context Manipulation Attacks	160
6.6	Limitations	163
6.7	Conclusion	164

7	Conclusion	167
7.1	Summary of Results	167
7.2	Future Work	169
7.3	Final Remarks	170
	References	171

List of Figures

4.1	Examples of effect of distance on surveillance footage quality.	41
4.2	Diagram of the scenario	43
4.3	Flow diagram of the detection algorithm.	46
4.4	Free space path loss simulation	48
4.5	Simulation of attack phase and proximity detection	52
4.6	Various example approach patterns	54
4.7	Simulation of a zig-zag approach	55
4.8	Detection ranges	56
4.9	Simulation of the detection algorithm	57
4.10	Deployment of the receiver	58
4.11	Drones used in the real-world experiments	59
4.12	Different flight patterns during the approach	61
4.13	Long-term test for a straight approach with the DJI Phantom 3 . . .	64
4.14	Short-term test for a straight approach with the Parrot Bebop . . .	65
4.15	Long-term test for a zig-zag approach with the DJI Phantom 3 . . .	66
4.16	Long-term test for a back-and-forth approach with the DJI Phantom 3	68
4.17	Attack phases for a back-and-forth approach with the DJI Phantom 3	69
4.18	Statistical tests for NLOS approaches over the roof	70
4.19	Long-term test for a NLOS approach around the house with the DJI Phantom 3	71
4.20	Approach patterns and equipment placement.	78
4.21	Histogram of the excess kurtosis values	84
4.22	ROC curves for various detection distances	85
4.23	AUC at different receiver locations	87
4.24	ROC curves for multiple receivers used in tandem	88
4.25	Message rates at different distances	90
4.26	Detection rates at different distances	91
5.1	Example of how events can affect different sensors	101
5.2	PEEVES system overview	106
5.3	Device deployment	108
5.4	Experimental setup	110

5.5	Window parameter grid search	115
5.6	Opportunistic attacker limitations	122
5.7	Type and quality of most distinctive sensor depending on its position	124
5.8	ROC curve for the spoofing attack detection	128
5.9	Average false alarm counts for masking attack detection	130
5.10	Score correlation	131
5.11	Success rates of the opportunistic attacker	134
5.12	System performance against a sensor-compromise attacker	135
6.1	Example of context-dependent signatures	143
6.2	Scenario for context-aware verification	145
6.3	Illustration of context-dependent drops in performance	149
6.4	Attack tree	152
6.5	Detection performance for spoofing attack detection	156
6.6	Context-aware performance gains	157
6.7	System performance against a sensor-compromise attacker	159

List of Abbreviations

AUC		Area under ROC curve
CAA		Civil aviation authority
COTS		Commercial off-the-shelf
DDoS		Distributed denial-of-service
DR		Detection rate
EER		Equal error rate
FAR		False alarm rate
FOV		Field-of-view
FPR		False positive rate
FPV		First-person-view
FSPL		Free-space path loss
GPS		Global Positioning System
IoT		Internet-of-Things
KNN		k-nearest neighbors
LOS		Line-of-sight
MAC		Media access control
NLOS		Non-line-of-sight
NTP		Network time protocol
PCFA		Conditional false alarm probability
RF		Radio frequency
RMI		Relative mutual information
ROC		Receiver operating characteristic
RSS		Received signal strength
RX		Receiver
SR		Success rate

SVM		Support vector machine
TPR		True positive rate
TX		Transmitter
UAV		Unmanned aerial vehicle
WSN		Wireless sensor network

1

Introduction

Contents

1.1	Motivation	2
1.2	Contributions	3
1.3	Outline	5

1.1 Motivation

With the advent of commercial smart home devices, the Internet-of-Things (IoT) paradigm is no longer a dream of the future. From smart locks to smart surveillance cameras to remote-controlled garage doors, IoT devices are starting to permeate consumer homes, enriching them with a wide variety of functions and information that is only a button press away.

Unfortunately, this new and fast-evolving market has pushed manufacturers towards quick development cycles that favor features and a fast time to market over security issues. This has led to an ecosystem where vulnerable devices are far-spread and often remain unpatched [1].

While the presence of cheap and possibly insecure devices in their homes puts consumers at risk, the pervasiveness of sensors also provides new ways to strengthen the security of these systems. As sensors can measure all kinds of physical effects occurring within a smart home and around it, they can thus be used as a means of providing additional situational awareness to the smart home. This principle has already been used to great effect for context-aware pairing [2, 3] and human presence recognition [4, 5]. In addition to enabling sensor-based security functions, this information can be further used to collectively alleviate some of the aforementioned risks and detect attacks that would otherwise go unnoticed.

However, given the different primary focus of the sensors, the data collected by these devices is inherently noisy and unreliable. The result of this can be that physical phenomena are not properly recognized and that there is a slew of false attack alerts which can reduce the usability of such a system. This problem can be especially egregious if it affects several correlating sensors simultaneously and could be exploited by a knowledgeable attacker. Furthermore, an attacker might additionally compromise the sensors tasked with providing this supplemental information to advance their goals.

Several approaches have been proposed to reduce the risks posed by insecure IoT devices. Most of these approaches either focus on device and software permission systems [6–8] or involve network traffic analysis [9–12] to detect unauthorized devices

or undesirable network behavior. But none of these approaches focuses on leveraging the abundance of already deployed smart home sensors to secure smart home systems.

To address this gap, this thesis investigates how the ubiquitous presence of sensors in smart homes can be leveraged to enhance the security of smart home systems and environments. There are already many different types of sensors readily deployed in modern smart homes, and we show that this pervasiveness of sensors can be used both to improve the security of these systems and to add further security features. In order to fully exploit their potential, these systems need to be able to use pre-existing sensor deployments and learn which sensors can be used for different purposes. As such, this thesis explores these notions through two systems with real-world application and analyzes the implications of different threat models on the security of the proposed systems.

1.2 Contributions

This section describes the contributions of this thesis to the field of system security. Although all the published work underpinning this thesis has been the result of joint work with others, I have included only papers for which I was the first author and main contributor. However, I was assisted by my co-authors in conducting the experiments and during the preparation of the publications.

- We first introduce the notion of privacy-invasion attacks perpetrated by unmanned aerial vehicles (UAV), commonly known as drones, and develop a drone detection system which uses radio signal strength metrics. This system can easily be implemented on repurposed and readily available hardware—making it accessible to consumers. The results of this work have been published at the 2017 Network and Distributed Systems Symposium (NDSS) [13]. An earlier version of this work made up part of my Master’s thesis; the published work is included here because it was further developed during my DPhil and because it lays the groundwork for the remainder of the thesis.

- We demonstrate that even receivers in non-advantageous locations can benefit our proposed system. We extend our previous drone detection system to better account for other types of moving transmitters by improving the set of statistical detection metrics. This work has been published in ACM Transactions on Privacy and Security (TOPS) [14] as an extension of [13].
- We first propose a system to verify spoofed events in event-driven smart home systems and evaluate it in a real-world office environment. We make the resulting dataset available to the research community to allow others to expand on our work. This work further introduces the notion of an opportunistic attacker who is able to predict attack opportunities to better time their attacks and has been published at the 2019 ACM Conference on Computer and Communications Security (CCS) [15].
- We extend our work on event verification through the additional consideration of masked events. We also evaluate our system against a particularly strong type of attacker who has the ability to compromise parts of the system during operation. We further demonstrate the benefits of a fusion scheme over a monolithic classifier against this type of attacker. This work has been published in ACM Transactions on Internet of Things (TIOT) [16] as an extension of [15].
- We investigate the risks and benefits of enhancing event verification systems with context-awareness. Fusion-based schemes in particular can benefit heavily from this additional information which allows them to evaluate the factors that might affect the performance of individual sensors. We show that it is paramount to consider the increased attack surface this addition may give to the attacker and demonstrate the importance of recursively verifying each contextual event.

1.3 Outline

The remainder of this thesis is structured in the following way:

- Chapter 2 provides an overview of related work and places this thesis in the context of the broader field. We describe the state of the art in smart home security in general, as well as context-based pairing, attacks on sensors, and attack-resilient sensor fusion.
- Chapter 3 gives the background for the two systems—drone detection and event verification—proposed in this thesis. We offer a detailed survey of research relating to these two systems, discuss the main metrics used for evaluation, and conclude with a set of research questions.
- Chapter 4 describes the design and evaluation of a system for the detection of UAV-based privacy-invasion attacks, while taking into account the challenges posed by cross-traffic of various static and ground-based moving transmitters. We derive a set of statistical metrics and evaluate their ability to identify drone movement and a drone’s proximity to a protected smart home environment in a real-world experiment. We show that our statistical metrics capture the difference between airborne and ground-based moving transmitters well, and are thus able to keep false alarms low and improve the usability of the system. The proposed system makes use of readily available hardware and only relies on the availability of received signal strength information. It is able to exploit sensors which have been deployed without drone detection in mind and can draw on the data of several devices to improve detection performance.
- Chapter 5 describes the design and evaluation of a system for event verification in event-driven smart home systems in order to detect both spoofing and masking attacks. We demonstrate how the proposed system can learn to detect and verify the occurrence of events without knowledge of the exact deployment details such as the nature of the events in question, or the location or physical properties of the sensors used for verification. We further evaluate

the system against two advanced types of attackers—opportunistic attackers and attackers with the ability to compromise sensors. A zero-effort attacker will often struggle to reliably attack ubiquitous sensing based event verification systems as the success of their attack depends on the timing of their attack coinciding with sensor anomalies which can lead to misclassification by the system. In contrast, an opportunistic attacker is able to wait for and predict these attack opportunities to improve their chances of success, whereas an attacker who can compromise sensors can create these opportunities themselves by manipulating the sensor data used for verification. We demonstrate the importance of considering these types of attackers in the context of leveraging ubiquitous sensors for security purposes.

- Chapter 6 shows how systems like the one introduced in the previous chapter can be improved through the use of additional context information which is often available in smart home environments. As sensor readings can be affected by various physical phenomena, detection performance of a sensor can vary widely in different contexts. We show how this information can be used to improve fusion-based event verification systems and what role recursive verification of contextual events plays in making sure that this performance improvement does not come at the cost of security when faced with a determined attacker.
- Chapter 7 finally summarizes the results of the work presented in this thesis, discusses future work and concludes the thesis.

2

Related Work

Contents

2.1	Security of Smart Home Systems	8
2.2	Compromised Devices	9
2.3	Mitigation of Compromised Devices	11
2.4	Context-based Pairing in Smart Home Systems	12
2.5	Attacks on Sensors	13
2.6	Attack-resilient Sensor Fusion	14

This chapter will contextualize the thesis within smart home security research and survey the related work for a variety of attack vectors and defense mechanisms.

2.1 Security of Smart Home Systems

Due to the increasing popularity of smart home platforms, there has been extensive research focusing on the security of smart home systems and devices in recent years. Fernandes *et al.* [17] investigated the popular SmartThings system and uncovered several vulnerabilities due to overprivileged smart home applications. These findings have prompted further research into smart home permission and access control systems [6–8, 18–26]. These systems control to what extent a device or smart application can access a resource, such as the physical capabilities of a device or stored user data, on the IoT network. Permissions can limit access to information flows between specific devices [8], depend on the activity currently being carried out by the user [18], be based on contextual information such as device states or environmental conditions [6], or constrain usage of a device to particular functionalities [19].

Other works have employed static analysis based on smart home application code to secure smart home platforms [23, 27–35]. Research in this area has provided solutions that range from identifying sensitive data flows [27] or events that can lead to unsafe system states [30] to checking the code for safety, security and functional properties [28] or physical inter-app connections [29]. IoTSafe [35] further combines static analysis with dynamic testing to tailor the discovery of physical side-channels to the current deployment.

Research has also considered the security of cloud-based trigger-action systems and the implications on home deployments [34, 36–42]. These types of systems further increase the attack surface of smart home setups as they rely on external platforms and their use can lead to unintentional information disclosure or integrity violations [36], can allow control of devices beyond the rules set out by the user in case of platform compromise [37], and the risks associated with their use are generally poorly understood by end-users [39].

The presence and normal operation of smart devices in a home environment has further been shown to affect the privacy of their users. Recent work has demonstrated how smart home activities and device event signatures can be inferred from network traffic, highlighting the privacy implications of wireless communication in smart homes [43–48].

However, just as critical are attacks against the devices themselves, as an attacker can take control of a device and cause it to operate in malicious and dangerous ways. Due to the low level of security of many smart home devices, there are a number of ways for an attacker to compromise them. An attacker can infect vulnerable devices that are connected to the internet with malware, leverage already compromised nodes in a network as a starting point for further attacks and infections, or use their physical proximity to an IoT network to attack vulnerable target devices over the wireless channel [49].

2.2 Compromised Devices

There have been numerous studies detailing vulnerabilities of individual smart home products and device types [50–55]. Another common attack vector are insufficiently secured IoT devices that are exposed to the internet. Recent prominent examples of malware targeting such embedded home devices were the Mirai [1, 56, 57], Hajime [58, 59], Persirai [60], and BrickerBot [61] botnets.

Mirai had a particularly large impact and gained access to webcams, DVRs, and routers by exploiting weak login credentials with bruteforce methods. Thus infected devices have been used as part of a botnet for various distributed denial-of-service (DDoS) attacks on popular websites. According to the authors of [1], this type of malware is continuing to spread because manufacturers in the IoT domain still do not follow basic security principles and proceed to sell weakly configured IoT devices. Recently, an overview of known DDoS-capable IoT malwares has been given by Spognardi *et al.* in [62]. This overview demonstrates the pervasiveness of and increasing risk posed by this type of malware.

In BlackIoT [63], Soltan *et al.* describe how IoT botnets have the potential to do far more damage than typical DDoS attacks. This is because well coordinated high wattage devices have the ability to disrupt power grids and cause large-scale blackouts.

But even if the smart home network is secure from attacks launched over the internet, it can get compromised. In [51], Ronen *et al.* describe an attack on smart lightbulbs that depends only on physical proximity and wireless connectivity. They describe how an attacker can drive through a city (*wardriving*) or even fly a drone in residential areas (*warflying*) to infect nearby devices. Moreover, the worm developed by the authors of this paper is able to spread to similar devices in the neighborhood. This highlights the danger of contagion: starting with only a few initial infections, devices all over a city could become infected, given a high enough density of vulnerable devices.

Non-IoT devices owned by the user are another possible attack vector to infiltrate a smart home. A malicious smartphone app has been used by Sivaraman *et al.* [64] to attack IoT devices from within the same network. They used an infected smartphone connected to a home network to scan for vulnerable IoT devices and report them to a remote attacker. Additionally, they exploit the fact that many routers or home gateways allow devices in the network to open arbitrary ports to the outside. In this way, the malicious phone app is able to expose vulnerable but otherwise inaccessible devices to the internet and hence to the attacker. Furthermore, as these devices are then connected to the internet they can be found by other opportunistic attackers. For the very purpose of revealing such vulnerable networks, there exist online search engines such as Shodan [65].

In conclusion, there exist a variety of ways to compromise IoT devices. Because these devices permeate smart homes, this insecurity demonstrates the need for smart home systems to be adversary-aware.

2.3 Mitigation of Compromised Devices

An important step towards mitigating the dangers of compromised devices is detecting the presence of these very devices. There has been an increased interest in identifying smart home devices and fingerprinting device behavior. Several solutions have been proposed that identify devices based on their network traffic fingerprint [9, 10, 66–68]. Other approaches have focused on determining normal device behavior in order to detect anomalous operation patterns [11, 69–76].

After a device has been compromised, it is important to learn more about how the attack has been executed in order to detect and address security issues. To enable such an attack analysis, multiple IoT forensics frameworks have been introduced [77–80].

A more proactive approach is taken by IoTPot [81]. IoTPot is a honeypot that exposes itself to telnet-based IoT malware to enable further analysis. The authors found four different families of IoT malwares capable of launching DDoS attacks targeting a variety of architectures. Honeypots such as this one can be used to distract the attention of attackers away from the real devices and make an early detection of current threats possible.

Like the malwares detected by IoTPot, many malwares use infected IoT devices as part of a botnet to launch DDoS attacks. AntibioTic [82] tries to alleviate this problem by pre-emptively infecting vulnerable devices with “good” malware. In this way, the system is able to clean already infected devices, warn the users of the existent threat, or pro-actively secure the devices against future attacks by changing login credentials or applying security patches. The application of these methods in the real world, however, is likely restricted to law enforcement agencies as it involves illegally exploiting software vulnerabilities and tampering with other peoples’ hardware.

2.4 Context-based Pairing in Smart Home Systems

In order to make sure that only authorized devices can enter a smart home network, there has been a substantial amount of research into the secure pairing of devices. See [83] for an overview of the large body of existing work. Most related to systems utilizing ubiquitous sensing for security are zero-interaction or context-based pairing schemes [2, 3, 84–92]. These types of pairing schemes have been proposed to make the pairing of devices more user-friendly by limiting the involvement of the user in the pairing process. Varshavsky *et al.* [84] developed a method that is based on fluctuations in the received signal strength (RSS) of Wi-Fi broadcast signals. In this manner, they use the existence of a shared radio environment as a means to check for co-location. In [85], Schürmann and Sigg propose a pairing scheme that uses fingerprints of ambient sound as a shared secret between co-located nodes. From these audio samples, they extract cryptographic keys using fuzzy extraction. Miettinen *et al.* [2] use luminosity as well as ambient noise to ensure co-location of devices. Additionally, they propose a key evolution algorithm to gradually increase the trust in the authenticity of a device over time.

A recent paper building on the idea of context-based pairing is Perceptio [3]. Perceptio relies on common sensor measurements to provide the entropy needed to derive symmetric keys. The authors of this paper argue that because devices present in a smart home have diverse sensing modalities, pairing protocols must consider this device heterogeneity in order to be usable. They make the observation that even if devices do not share a common sensor type, co-located devices can be affected by the same physical events in different ways.

All the schemes described above have in common that they make use of the sensing capabilities of devices within the network to measure wireless signal strength or perceive ambient, physical phenomena.

2.5 Attacks on Sensors

In smart home systems, sensor readings are typically implicitly trusted. However, it has been shown that attackers can exploit physical properties of a sensor to trick it into becoming an unintentional receiver and propagator of unwanted, malicious signals [93]. A good overview of attacks on sensors and possible defenses is given by [94]. These types of attacks have been demonstrated for a wide range of sensors [95–110], some of which can commonly be found in smart home devices such as inertial sensors [100–103], microphones [104–109], or temperature sensors [110].

In GhostTalk [104], Kune *et al.* demonstrated that electromagnetic interference can be used to inject signals into analog sensor systems. They demonstrated their attack by injecting audio into microphones of consumer electronics and disrupting the normal operation of an implantable cardiac device. These attacks were successful at distances of up to two meters. They state that by adding appropriate analog shielding and filters, the vulnerability of commodity devices to these kinds of attacks can already be greatly reduced. Additionally, they propose a metric to measure the signal contamination caused by the signal of the attacker.

DolphinAttack [105] is another attack that can inject audio into microphones unnoticed. In contrast to GhostTalk, DolphinAttack uses the non-linearity of microphone circuits to inject inaudible voice commands via the ultrasound channel. The authors of this paper demonstrate successfully that these voice commands are understood and executed by current voice controllable systems like Siri and Google Now.

Another form of acoustic attack has been demonstrated with WALNUT [103]. The authors of this paper show how intentional acoustic interference can be used to spoof accelerometers. They show in experiments that 65% of the twenty tested models are vulnerable to output control through this attack. The attack proves to be successful even when the built-in speaker of a smartphone is used to take control of the phone’s accelerometer output.

Motivated by attacks targeting sensors, several defense mechanisms have been developed [91, 104, 105, 109–116].

PyCRA [111] introduces a physical challenge-response authentication scheme that protects active sensors against such attacks by using random yet deliberate physical stimulation of the sensors with subsequent behavior analysis. They demonstrate the feasibility of their system in three case studies: magnetic encoder attack detection, magnetic encoder attack resilience and RFID eavesdropping detection. However, Shin *et al.* [113] demonstrated that PyCRA can be easily bypassed by an attacker.

To overcome this issue and in order to also enable passive sensors to detect electromagnetic interference attacks, Zhang *et al.* [116] have proposed a method that switches the sensor in question off in an unpredictable manner. The microcontroller monitoring the sensor can then detect an adversarial signal if it measures a sensor output voltage that deviates from a known constant value (*e.g.*, 0V) during this off-period.

To thwart sensor replacement attacks, the authors of NoiSense [112] suggest the use of sensor fingerprints based on a sensor’s noise profile. Sensor fingerprints are learned and tested with a Support Vector Machine (SVM). They show in experiments conducted with 44 sensors of six different types that their approach is able to uniquely identify sensors with up to 97% accuracy.

The readings of sensors can often be manipulated by physically present attackers. Defenses proposed by the papers in this section can be used to complement the security of systems which build on ubiquitous sensing.

2.6 Attack-resilient Sensor Fusion

The possibility of falsified sensor data has been recognized in areas that rely heavily on sensor data such as cyber-physical systems (CPS), wireless sensor networks (WSN) and co-presence detection.

In CPS this is typically mitigated through resilient state estimation [117–121] and physical anomaly detection techniques [122–126] that both exploit that the underlying physical processes of these systems are well understood due to their closed-loop designs. To combat more stealthy attacks that are able to evade anomaly detection, Ivanov *et al.* [122] propose to use dynamic fusion intervals that allow

the system to stay resilient to such attacks as well as to transient faults. In smart home systems, however, events happen sporadically and the specific correlations between actuators and different sensors are unknown at design time. More recently, deep learning methods have been proposed for CPS security applications [127]. Yet, there have already been studies showing how these detection systems can be evaded by a determined attacker [128].

According to Illiano *et al.* [129] research on malicious data injection in WSNs can be grouped into anomaly detection techniques [130–135] and trust management approaches [136–138]. In [135], the authors state that typical algorithms to detect anomalies in sensor network management are not expected to perform well in the presence of an adversary, as they are not designed to cope with colluding nodes. To close this gap, the authors propose a method to detect spoofing and masking of events in sensor networks based on a spatial correlation technique. While this method is shown to work well in simulations and on a real-world dataset of seismic vibrations, it is not applicable in a typical smart home context. This is due to the fact that it is designed with large-scale networks in mind and because it requires that the locations of sensors are known. Furthermore, it focuses on a single sensing modality and is as such not usable with a more diverse set of sensors.

As introduced earlier, methods using the co-presence of devices for context-based pairing rely on the measurements of environmental sensors to establish the necessary entropy to create a common secret. Shrestha *et al.* [91] showed how readings of different acoustic, radio-frequency and physical environment sensors can be manipulated to gain an advantage in defeating context-based co-presence detection methods. In this paper, the authors compare two different approaches to fuse the different sensors: feature-fusion, which feeds the features of all sensors into a single classifier, and decision-fusion, where classifiers are trained on sensor subgroups before the decisions of these classifiers are aggregated. In their evaluation, the authors observe that fusing the used sensors on a decision level can reduce the attack success rates when dominant sensors are manipulated. However, this can come at the cost of usability as weaker sensors might increase false negative

rates. The observation that fusion methods can introduce this tradeoff between reducing the overall attack success rate and increasing the reliance on weaker sensors informs our approach to fusing sensors in a robust way to overcome the risk of individual compromised sensors.

3

Background

Contents

3.1 Drone Detection and Countermeasures	18
3.1.1 Drone Detection	18
3.1.2 Countermeasures	22
3.2 Event Detection and Verification	22
3.2.1 Event-driven Smart Home Systems	23
3.2.2 Event Detection in Sensor Networks	25
3.2.3 Activity Recognition in Home Environments	27
3.2.4 Event Detection for Security	30
3.3 Metrics	31
3.4 Research Questions	33

This chapter will discuss related work for the two real-world systems that we propose in this thesis—drone detection and event verification in event-driven smart home systems—and thus provide the background needed to investigate the use of ubiquitous sensing for security. Furthermore, we will introduce the main performance metrics found in related work and we will justify the metrics used for evaluation in this thesis. Finally, we will conclude this chapter with the set of research questions guiding the work in this dissertation.

Table 3.1: Comparison of drone detection approaches. Hybrid systems can combine the benefits of several approaches and are able to reach high detection performances at the expense of higher costs. This table has been adapted from [139].

Sensor modality	Range	Cost	Detection performance	LOS required	References
Acoustics	Low–Medium	Low	Low–Medium	✗	[140–144],
IR camera	Medium	Medium	Medium	✓	[145]
LiDAR	Medium	High	Medium–High	✓	[146]
Radar	Very high	Very high	Medium–High	✓	[143, 147–149]
RF	Medium–High	Low–Medium	Low–High	✗	[13, 144, 150–152]
Video camera	Medium	Low	Medium	✓	[143, 144, 153]

3.1 Drone Detection and Countermeasures

In this section, we will focus on the topic related to the first of our proposed real-world systems: drone detection. We will first survey existing UAV detection methods, discuss their applicability in smart home settings and subsequently give a short overview of available defense mechanisms.

3.1.1 Drone Detection

Drone detection can be achieved through a variety of methods. Prior work has focused on improving the underlying techniques of each method, but they still all have notable limitations to them. Many are very expensive to deploy and they often require specialist expertise to operate. In commercial systems, different methods are typically combined into hybrid approaches—leading to an additional increase in costs and complexity of the systems. We describe below key results that show the behavior of different methods; a more in-depth survey is presented in [139]. Many of the surveyed papers provide no performance metrics and those that do often rely on different measures than others, making direct comparisons challenging. Nevertheless, we present performance metrics where available. See Table 3.1 for an overview of the detection approaches discussed in this section.

Traditionally, radar systems have been and are still being used for aircraft detection both for military and aircraft control purposes. Radars, however, have to be designed with a given aircraft size in mind, as the frequency of the transmitted radio waves has to be chosen accordingly. Therefore, radar systems used for

conventional aircraft monitoring can not be used for drone detection because drones are too small to be picked up by those systems. To address this issue, high-frequency radars in the GHz range have been developed [147, 148]. While such systems have been shown to be able to distinguish birds, drones and small planes with a probability of correct classification of the order of 95% in an ideal setting [149], they are prone to exhibit performance degradation in adverse weather conditions. But the main issues which stop the average person from deploying any radar system in their home are the large expenses for the necessary equipment, as well as the requirement for trained operators and the appropriate transmission licenses.

In contrast to radars, image and video analysis based on the appearance and movement patterns of drones has the potential to provide a more affordable way to detect UAVs, with an average precision of 75.1% quoted in [153]. However, methods based on the appearance of drones are challenged by the regular commercial releases of new drone models in different sizes and shapes, while motion-based techniques often struggle to differentiate between drone and bird movement [142]. In addition, line-of-sight is generally required for the visual detection of a drone. The detection performance can further be affected by the level of lighting and other visibility conditions such as fog or smoke. Infra-red light, on its own or in tandem with the visible light spectrum, can be used to improve the detection performance in low-light situations. However, this method is still prone to confusing drones with birds and struggles to recognize drones if the drone’s surface temperature is close to the ambient temperature [145]. Irrespective of the type of light observed, real-time image processing is computationally expensive and not readily incorporated into existing consumer devices. This is even more true for LiDAR systems, which—despite showing promising detection results—still require line-of-sight conditions and a substantial investment in expensive specialist equipment [146].

Many of the limitations of visual detection methods can be avoided by using microphone arrays to acoustically detect drones. These types of detection systems are effective even at night and do not require line-of-sight contact with the drone [142], achieving a 99.5% probability of detection with a false positive rate of 3% for ranges

smaller than 600m in good conditions [141]. Nevertheless, these systems are prone to acoustic variants of some of the same problems that plague visual detection methods. Bad weather can lead to distorted sounds and noisy environments can severely hamper detection performance. Acoustic detection techniques also rely on the existence of a thorough and up-to-date database of drone rotor sound signatures. This means that these systems have to be constantly updated with new signatures to stay relevant. Furthermore, while it is possible to build these systems using off-the-shelf components, even examples that are designed to be cost-effective like the low-cost acoustic array system by Case *et al.* [140] are unaffordable for the general public; this particular example requires an array of 24 microphones and cost \$3768 to build.

Acoustic arrays and video cameras can be combined with radar or radio frequency (RF) based detection methods to create hybrid approaches such as CSUAV [143] and ADS-ZJU [144]. These hybrid approaches can benefit from the advantages of all their sub-modules. The authors of [144] report individual detection probabilities of 76.9%, 98.1% and 88.2% for the audio, video and RF sub-modules, respectively, when the drone is at a distance of 50m. The corresponding false alarm probabilities are 2.2%, 1.1% and 2.8%. But while the combined use of these approaches improves the detection probability to 99.3% with a false alarm probability of 5%, they also come with a substantially increased complexity and cost of the system. It is also noteworthy that the authors attribute the low false alarm probability of the video sub-module to a lack of occlusions during the experiments.

Due to the widespread use of wirelessly communicating devices in home environments, methods which use RF signal processing to detect drones are of particular interest when considering domestic settings. RF receivers can be used to locate the origin of an RF signal and can thus detect a drone by virtue of it being an airborne transmitter. But methods based on localization generally require multiple receivers and very precise time-synchronization between those receivers [154]—which again drives up deployment costs.

To facilitate the use of ubiquitous sensing for drone detection, our focus lies on leveraging inexpensive consumer hardware and reusing existing smart home systems.

Previous work in this area has shown that protocol signatures and media access control (MAC) address prefixes can be used to detect the presence of drones [150], if such information is accessible. However, these methods rely on either the use of unencrypted wireless communication or access to known manufacturer lists. A contrasting example is offered by work looking into the use of low-level physical effects, observed in an RF signal, to identify drones [151]. This approach is able to recognize drone transmitters based on movement characteristics with an accuracy of 96.5%, a recall of 97% and a precision of 95.9% at a distance of 10m, but it requires access to the raw radio signal. This in turn requires either the deployment of a purpose-built detector or the use of software-defined radio implementations, which make it expensive compared to a system using only commonplace consumer devices. Crucially, in both of these cases, these kind of systems can not differentiate between a neighbor turning on a drone in their own house, a benign flight taking place in the near vicinity, and a trespassing drone intended on violating someone's privacy. To overcome this limitation, recent work has demonstrated that it is possible to confirm if the camera of a drone is recording a specific property. Nassi *et al.* [152] built a system which deliberately injects patterns into the network traffic of a drone through the use of visual stimuli. These visual stimuli are picked up by the drone's camera and send over the live video feed to the drone's controller. By verifying the existence of those patterns in the observed network traffic, the system is able to determine if the property is under surveillance by the drone. This system achieves perfect detection rates with a false positive rate below 10% at a distance of 30m. However, it requires a specialized and purpose-built system, and relies on the use of variable bitrate video encodings by the drone.

Nowadays, a significant number of drone detection systems are commercially available. But even the more accessible systems, such as the similarly named Drone Detector¹ and Drone-Detector², are more targeted at specialist use cases and not designed with domestic settings in mind. Other manufacturers such as Thales³,

¹www.dronedetector.com

²www.drone-detector.com

³www.thalesgroup.com/en/squire-ground-surveillance-radar

SAAB Group⁴ and Chess Dynamics⁵ built their systems with a focus on military use. Most of the aforementioned systems have in common that they are meant to protect high-security facilities rather than home environments. Therefore, they use expensive, specialist equipment instead of ubiquitous consumer hardware.

3.1.2 Countermeasures

After a maliciously acting drone has been detected, the defending system or user may want to launch countermeasures to stop the attack. Previously proposed countermeasures range from downing the drone with a shotgun or a flying net [155] to capturing it with a live eagle [156] or taking over the flight-controls through exploits [157]. Other options include GPS spoofing [158, 159] and an array of denial-of-service attacks such as control-signal jamming [159], Wi-Fi de-authentication attacks [150], or interference with on-board gyroscopes via acoustic resonance [100]. A more accessible approach that can be used in smart homes outfitted with smart blinds is as simple as automatically shutting the blinds when a drone is observing the house, which prevents useful video footage from being gathered. This can be easily implemented in event-driven smart home systems, which are systems in which events (*e.g.*, device state changes or the detection of a nearby drone) reported by one device trigger actions (*e.g.*, shutting the blinds or sending an alert email) by other devices [160].

3.2 Event Detection and Verification

In this section, we will introduce the related work for our second proposed system, which enables the aforementioned event-driven smart home systems to verify that reported events have actually taken place.

⁴<https://saab.com/air/sensor-systems/ground-based-air-defence/giraffe-amb/>

⁵<http://www.chess-dynamics.com/auds/>

3.2.1 Event-driven Smart Home Systems

A security analysis of event-driven smart home systems was first conducted by Fernandes *et al.* [17]. In their work, they focused on the SmartThings [161] framework by Samsung as they identified it as the most wide-spread system of this kind. The authors discovered design flaws of the SmartThings event and permission system that led to smart home applications (or SmartApps) being able to take advantage of overprivileged access to events of other devices. In particular, they showed that unprivileged SmartApps were able to read all the events of any device. The only requirement for this attack was the possession of the device identifier for the target device. These identifiers were not kept secret and as such were available to any SmartApp with read access to an event of the device in question. Hence, any SmartApp with access to one event of a device could also gain read access to any other events provided by the device. Furthermore, they showed that SmartApps are able to spoof and inject any event into the SmartThings event system using the appropriate device identifier. The authors exploited these flaws in several example attacks to snoop the PIN codes for a smart lock, to disable the vacation mode of a smart house system and to trigger false alarms of a smart home security system. These attacks highlight how spoofed events can be used to make a smart home less secure.

Jia *et al.* [6] warned of malicious apps in appified IoT environments that can give an attacker access to devices. In their work, they focus on flaws in the permission systems of IoT frameworks and provide a new context-based permission system. This context-based approach is motivated by the observation that actions in smart homes are often desired only in certain situations, *e.g.*, a SmartApp that gets the permission to open the window when the temperature has exceeded a threshold should not be able to perform this action under different trigger conditions (*e.g.*, when the smart home switches to a “Sleep mode”). By adding the context—in this case the triggering temperature event—to the permission granted by the user, ContextIoT manages to alleviate this problem. An interesting takeaway from this paper is the interconnectedness of smart home devices through events. This means

that a compromised device can mislead another device to execute actions by merely lying about its own events. In the example from the paper, the attacker could achieve their goal of opening the window by compromising the temperature sensor. The temperature sensor could then force the window to open by sending false reports of high temperatures.

The finding made by ContextIoT that device actuations can lead to unintentional consequences in event-driven systems has also been replicated for trigger-action rules that make use of external trigger-action platforms [22], and for rules that include hidden inter-app interactions through physical channels [29]. These issues have been addressed through more fine-grained permission systems [6, 18], better logging for forensic purposes [77], and static analysis of smart applications [22, 28–30] to detect possible physical channel interactions. While these systems help to reduce the risk that these interactions pose, they do not protect against cases where an existing and desired trigger-action rule is being exploited and they can not protect against a smart home device that is itself compromised.

To this effect, Ronen *et al.* introduced extended functionality attacks as a new type of attack that can be launched from compromised devices such as smart lightbulbs [162]. They subdivided these attacks further into attacks where the original functionality of the device is either: ignored, reduced, misused or extended. The paper focuses on the last category and demonstrates two specific attacks on smart lightbulbs. One attack involves triggering epileptic seizures in susceptible victims using strobed light, the other is the creation of a covert channel through on-off keying by switching the dimming setting between two settings not differentiable by humans. Although the data rate of this simple approach was quite low (10KB per day), the authors were able to successfully transmit data from a distance of over 100 meters.

The observation that devices can ignore, reduce and misuse their event-producing functionality motivates our proposed event verification system, which makes sure that events can be verified through other means. To verify the occurrence of reported events in the proposed system, we have to be able to detect these events

using the aggregate sensor data. Event detection in sensor networks and activity recognition in home environments are two research topics that demonstrate how to approach this objective in a variety of scenarios.

3.2.2 Event Detection in Sensor Networks

In wireless sensor networks (WSN), events are often considered as a special form of outlier in the sensor data. The main distinguishing factor between events and noisy measurements or sensor faults is that event measurements are likely to be spatially and temporally correlated, while noisy measurements or sensor faults are likely to be stochastically unrelated [163].

A range of methods for event detection have been proposed, approaching the problem from different angles. Initially, many of these approaches used threshold-based event detection techniques to check whether the sensor measurements exceed predefined limits [164–166]. The benefit of these approaches is that they are simple and lightweight to implement. However, setting the right threshold values can require expert knowledge. Furthermore, as many types of events show more complex patterns and are less likely to be sufficiently captured by simple thresholds, machine learning algorithms have been identified as useful methods for efficient event detection of more elaborate events [167].

In some works, a mixture of the two approaches is used. For example, Yu *et al.* [168] investigated forest fire detection in wireless sensor networks using a combination of threshold-based and neural network based event detection. First, the sensor nodes perform a threshold-based event detection locally. They then send their report to a managing node which uses a neural network to make the final decision about the occurrence of a forest fire. The authors showed the feasibility and efficiency of their approach in simulations.

Wittenburg *et al.* proposed a system for distributed event detection in wireless sensor networks [169] in which the decision about the occurrence of an event is made locally based on its own measurements and the data of neighboring nodes. The classification is then done on the aggregate in each node, using node-specific

prototype vectors. An event is detected if the feature vector is close enough to its nearest prototype vector in terms of Euclidean distance. The system was tested by mounting a hundred nodes with accelerometers along a construction site fence to create a fence surveillance system. The four monitored events in this experiment were “Shake”, “Kick”, “Lean” and “Climb”. To train the system, the authors collected data of fifteen events for each event type at one of the nodes. The test data consisted of fifteen events of each event type at three different locations, one being the training location (new data was collected; the training data was not reused). The system achieved an overall accuracy of 74.8% on the data of this experiment.

Bahrepour *et al.* [170] proposed a two-level approach that uses data fusion of several distributed sensors to stay resilient to the deployment area and to the dynamic nature of the WSN itself. They first trained both Feed Forward Neural Networks and Naïve Bayes classifiers for each sensor. Using these classifiers, each sensor then decided locally if an event was occurring, propagating the decision in case of a detected event to a second level classifier, trained on the decisions of the individual sensors. They evaluated their approach on the NIST smoke alarm dataset [171] and showed that it was able to detect fire events with high accuracy. For both types of classifiers the two-level approach increased accuracy, with the Naïve Bayes classifier showing more significant gains due to worse performance in the one-level case.

In another paper by the same authors [172], the base level classifiers are replaced by decision trees, and the second-level classifier is replaced by a reputation-based voting method that is used to reach consensus between the sensors. For this reputation-based voting method, the base level classifiers first exchange their local classification result. These classifiers evaluate their neighboring nodes by comparing their own measured value with the the values of the neighbors. Based on a threshold chosen by context, they judge if the difference between those values is too large. In this case, they give the neighbor a negative vote, otherwise they support it with a positive one. These votes then get sent to the node responsible for the fusion, which derives a global reputation based on the local reputations. Using these global

reputations, sensor weights are computed to make the final decision. This paper uses the same dataset for evaluation as the previous paper. In their evaluation they compare reputation-based voting against majority-voting schemes. They show that reputation-based voting outperforms majority-voting in most cases.

There is very little research on the effect of compromised nodes on event detection. In [135], the authors state that typical algorithms to detect anomalies in sensor network management are not expected to perform well in the presence of an adversary, as they are not designed to cope with colluding nodes. To close this gap, the authors propose a method to detect spoofing and masking of events in sensor networks based on a spatial correlation technique. While this method is shown to work well in simulations and on a real-world dataset of seismic vibrations, it is not applicable in a typical smart home context. This is due to the fact that it is designed with large-scale networks in mind and because it requires that the locations of sensors are known. Furthermore, it focuses on a single sensing modality and is as such not usable with a more diverse set of sensors.

The papers in this section show different approaches to event detection and provide interesting directions to explore in smart home systems. However, in contrast to current smart home systems, event detection systems in sensor networks are typically densely deployed, large-scale and single-purpose systems. We therefore additionally explore activity recognition in home environments.

3.2.3 Activity Recognition in Home Environments

In recent years, activity recognition in home environments has become another popular field of research that aims to use the combined knowledge of several sensors to detect the occurrence of events.

“Activity recognition in the home using simple and ubiquitous sensors” [173], is a seminal paper that introduced the use of many state-changing devices to infer activities carried out by two test subjects in their own homes. In this paper, a multiclass Naive Bayes classifier was used to determine 35 different occupant

activities such as “Preparing lunch” or “Leaving for work” based on binary state-changes sensors (reed switches) installed on doors, cupboards, switches and everyday household devices. In the apartment of the first test subject, 77 state-change sensors were installed, whereas the number of sensors in the second subject’s apartment was 84. These sensors collected data for 14 days in each apartment. Although the accuracy of the system presented in this paper was quite low, it has given birth to a new field of research and has since inspired a wealth of additional research [174–177].

In “Sensing from the basement” [178], Fogarty *et al.* use several low-cost sensors placed in a home’s water distribution infrastructure to infer activities in the household. To study the feasibility, they deployed microphones near waterpipes into a real home for six weeks. The data was labeled manually with water flow related events such as “Shower” and “Toilet flush”, and support vector machines were trained on each sensor. To examine the feasibility of their approach, the authors of the paper deployed sensors into a home occupied by two people for six weeks. The detection accuracies of the system ranged from 73% to 100% depending on event type.

ViridiScope [179] is a system that monitors domestic power usage indirectly through ambient sensors. These sensors are magnetic sensors, microphones and light sensors. Because the standard deviation of the magnetic field change strongly correlates with the power consumption, magnetic sensors can be used to estimate the current power usage. The other two sensing modalities are used to infer the internal power state of the devices. For example, a microphone can detect when the refrigerator is running, whereas a light sensor can detect when the door of the refrigerator has been opened. Using the average power usage for each state, the system then solves a numerical optimization problem to determine the approximate power usage. It was shown through experiments in a two-person apartment that the system can estimate the end-point power consumption within 10% error.

ElectriSense [180] is a system that detects operating home devices based on the fact that power supplies generate high frequency electromagnetic interference that is measurable through the home’s power wiring. The system requires only a single custom Power Line Interface module plugged into an ordinary power socket. The

authors tested their proposed system in one house for six months, and additionally conducted several one-day experiments in multiple other homes. The evaluation of these experiments showed that the system was able to classify and distinguish four different devices using a KNN classifier with a mean accuracy of 93.82%.

Several other papers have researched indirect sensing of specific home events in similar ways. ThermoSense [181] for example, uses an array of low-cost thermal sensors along with a passive infrared sensor to measure room occupancy for heating, ventilation, and air-conditioning control. GasSense [182] is able to identify the source of gas use (for example a water heater or a fireplace) by using a microphone near the gas regulator to detect the acoustic signatures of gas events. Finally, HydroSense [183] can detect the usage of individual water fixtures within a home by using a single water pressure sensor at a water valve.

SyntheticSensors [184] is a more recent paper which aims to provide general-purpose sensing. In this paper, synthetic sensors are introduced as an abstraction to events in a home environment to improve the usability of such a system. The system is based on a single sensor board with nine sensors covering twelve different sensing modalities. Individual synthetic sensors are then represented by a base-level Support Vector Machine (SVM), and a global multiclass SVM was trained on the output of all the base-level SVMs. Examples for the 38 synthetic sensors considered in this paper are events that also typically appear in smart home systems, such as “Kettle on”, “Door closed” or “Dishwasher running”. The system was tested for seven days at five different locations and achieved an average accuracy of 96%. It is noteworthy that the decision for most of the detected events was predominantly based on the data by two sensors, the accelerometer and the microphone.

The survey of related research in event detection and activity recognition shows that it is common to use classifiers to fuse the data or decisions of individual sensors. However, event detection and activity recognition systems are typically not designed with an adversary in mind. We therefore expect these systems to be vulnerable to an attacker that has managed to compromise one or more of the sensors included in the decision process. Especially when one sensor is more reliable than the others,

it might be possible that it, if compromised, can influence the decision strongly, even when its decision is not supported by the other sensors.

3.2.4 Event Detection for Security

Recent research has exploited the fact that smart home events have physical effects that can be measured through various sensing modalities. The previously mentioned Perceptio [3] uses common sensor measurements to provide the entropy needed to derive symmetric keys for context-based pairing. The authors of this paper make the observation that even if devices do not share a common sensor type, they can be affected by the same physical events in different ways. For example, when in use, a coffee machine has a distinct sound fingerprint that can be picked up by a microphone. If the coffee machine is connected to a power meter, that device can detect the power usage of the coffee machine. Although those two sensors might measure very different patterns and even register the event at different times, the timespan between two subsequent events is the same. Perceptio uses this inter-event timing information to construct fingerprints for each event type witnessed by a sensor. The event detection that is the basis for the fingerprint extraction method first uses a threshold to distinguish event signals from noise, before grouping similar events using K-Means clustering. These fingerprints are then used to provide the entropy for its pairing protocol. In contrast, our proposed event verification system leverages the simultaneous effect of a physical event on different sensors to verify that said event has occurred.

A further motivating example for the need of secure event detection is provided by Orpheus [124]. Orpheus introduces an event-aware finite-state automaton as a new program behavior model to detect data-driven attacks on control programs running on embedded devices. This behavior model links the program control flow paths with measurable effects on the environment. Data-driven attacks are then detected if physical events and the state transitions of the automaton are inconsistent. For this detection, Orpheus relies on a trustworthy event-driven sensor system to provide the measurements.

A system with a similar goal to our proposed event verification system is HoMonit [71], which uses network traffic based event fingerprints to detect event spoofing from misbehaving smart applications. HoMonit builds automata that model the behavior of smart applications and then extracts event signatures from wireless smart home traffic. In combination, these methods enable the detection of misbehaving smart applications if the expected behavior deviates from the actual behavior inferred from the network traffic. However, this does not prevent event spoofing when the attack originates from outside the smart hub, *e.g.*, because of a compromised smart home device. In this case, the event network signature will be present although the triggering event has not occurred.

The work outlined above demonstrates the need for an event verification system such as the one proposed in this thesis: a system that is both able to use the situational awareness provided by other devices in the environment and is resilient to compromised devices.

3.3 Metrics

As shown in the previous two sections, related work uses a variety of performance metrics to report results. In the following section, we outline a number of commonly used metrics, and discuss which ones we rely on for the remainder of this thesis.

True positive rate (TPR) is the fraction of samples that are correctly identified as belonging to a specific category when faced with a binary decision. In the context of attack detection, this is often referred to as the *detection rate (DR)*, or the fraction of actual attacks correctly identified as such.

The *false positive rate (FPR)* similarly gives the ratio of samples that do not belong to this specific category but are erroneously classified as such. For systems that are meant to distinguish attacks from benign samples, false positives result in false alarms, and the FPR is hence often called the *false alarm rate (FAR)*. In this thesis, we use the terms DR and FAR, instead of TPR and FPR.

Accuracy is often used to provide a combined measure of FAR and DR, as it provides the fraction of correctly classified samples. Combined measures such as

this are convenient as they allow comparisons between systems based on a single measure. However, accuracy does not distinguish between different types of errors and is therefore misleading when the classes are heavily imbalanced. This means that if the system always predicts that a sample belongs to the more frequent class—typically this means that no attack has occurred—this might lead to a very high accuracy. But such a system would be entirely ineffective, as it would be incapable of detecting any sample of the less frequent class.

Precision can be used instead of FAR, if false alarms have a large impact on the perceived performance of a system. It is defined as the ratio of detected attacks out of all the samples classified as an attack by the system. Precision is typically used in conjunction with *recall*, which is another name for DR.

The *F1 score* is a measure that combines precision and recall by equal amounts. Due to its use of precision instead of FAR, the F1 score is a combined measure that is less affected by imbalanced class distributions than accuracy is. However, it still gives equal weight to both precision and recall, meaning that it can be inappropriate to use for systems that put a higher emphasis on one or the other.

The *receiver operating characteristic (ROC) curve* is a plot that is used to show the tradeoff between DR and FAR. It shows the DR and FAR of a system at different operating points when the threshold governing the decision of whether a sample belongs to one class or another is varied.

Area under the curve (AUC) is a measure that summarizes a system's ROC curve, and is frequently used to compare different ROC curves with each other.

When a system is based on ubiquitous sensing, it is important that the system does not inconvenience the user with an excessive amount of false alarms. As these systems are supposed to be operated continuously, they should be expected to deal with benign samples most of the time, with attacks being far and few between. Due to the base-rate fallacy [185], this means that the FAR has to be extremely low for the system to remain usable, as even seemingly low FARs might result in many false alarms per day—leading the user to abandon the system. In this thesis, we therefore focus on the detection rate at false alarm rate zero as a comparative

metric. To compute this metric, we first tune the system to an operating point at which it never causes any false alarms and then give the resulting detection rate. While this can be difficult to achieve in practice, it is a helpful mechanism to analyze how much added security can be achieved by a system that reuses already deployed sensors, without inconveniencing the user.

A variety of measures are used in related work and there is a lack of consistency across the field: papers proposing systems similar to ours measure their performance using a range of different metrics, and a metric heavily relied on by one paper might not be reported by another (*cf.*, Section 3.1.1). This is due to the fact that these papers often have different objectives—some might not even consider an adversary—and therefore may use different metrics to reflect this. Therefore, it can be challenging to compare our work to solutions by other papers. As mentioned before, our goal in this thesis is to provide extra security without any additional expense or inconvenience to the user. As such, we put a higher emphasis on avoiding false alarms than related work that, for example, considers a high-security setting, as they might be more concerned with not missing any attacks. Nevertheless, we additionally compare our proposed systems to analogous systems using the same metrics where appropriate, and at the same operating point wherever possible.

3.4 Research Questions

Following the motivation set out in Chapter 1 and the insights gained from related work in Chapters 2 and 3, this thesis addresses the research questions stated below:

- **RQ1:** How can sensors that are already available in smart homes be used to enhance the security of the environment they are in?

The ubiquity of sensors in smart environments provides an opportunity to use their collective awareness of physical occurrences to draw conclusions about what is happening around them. In order to exploit this untapped potential, there is a need to investigate how smart home systems can use this sensor information without impacting the primary function of the sensors. Ideally,

this allows the system not only to mitigate current security issues, but also to provide entirely new security functions that were previously not possible without specialized hardware.

- **RQ2:** How does the placement of these sensors affect their ability to provide this security function?

Smart devices are placed by users within their homes at locations determined by the devices' primary functions. It is therefore essential that a system that uses the sensors of these devices is able to accommodate arbitrary device deployments. The system should be able to leverage devices as they are deployed and should not require devices to be moved or changed in a way that affects their primary function or inconveniences the user.

- **RQ3:** How can a system learn such security functions autonomously?

In order to be able to adapt to a specific environment and sensor deployment, such a system needs to learn *in situ* how it can carry out its security functions. To avoid burdening the user with this task of tailoring the system to their environment, the system needs to be self-reliant and able to adjust independently.

- **RQ4:** What capabilities might an attacker gain when these sensors are used for security purposes in this manner?

An attacker who knows that certain sensors are used for security purposes might exploit this knowledge to further their goals. Furthermore, the attacker might seek to compromise specific sensors in order to overcome the system's security mechanisms. It is therefore important to carefully consider the implications of reusing sensor data in a security context when faced with powerful adversaries.

- **RQ5:** How do contextual changes affect sensors' abilities to fulfill their security functions, and how can contextual information be used to mitigate potentially adverse effects?

Sensor performance can be significantly affected by changing conditions in the environment. There is therefore a need for systems using ubiquitous sensor data to understand the contexts in which this sensor data can be relied upon, as well as the contexts in which the data may be unreliable. Hence, these systems should employ additional contextual information about the environment in which they are deployed in order to decide how to use the available sensor data, thus enhancing the security of the system.

4

Real-world Drone Detection Using Smart Home Devices

Contents

4.1	Introduction	38
4.2	Background	42
4.3	Adversary Model	44
4.4	System Model	46
4.4.1	Pre-processing	47
4.4.2	Statistical Tests	47
4.4.3	Attack Analysis	49
4.5	Standard Deviation-based Method	49
4.5.1	Attack Analysis	51
4.5.2	Detection Range	53
4.5.3	Approach Patterns	53
4.5.4	Detection Algorithm	55
4.6	Preliminary Experiment	58
4.7	Preliminary Evaluation	60
4.7.1	Straight Approach	61
4.7.2	Zig-zag Approach	65
4.7.3	Back-and-forth Approach	67
4.7.4	NLOS Approach Over the Roof	67
4.7.5	NLOS Approach Around the House	69
4.7.6	Detection Ranges	69
4.8	Discussion of Preliminary Results	72
4.8.1	Results	72
4.8.2	Observations	73
4.8.3	Deployment Considerations and Mobile Transmitters	73
4.9	Experiment with Mobile Transmitters and Interfering Traffic	74

4.9.1	Non-drone Approaches	75
4.9.2	Static Transmitters	75
4.9.3	Drone Approaches	76
4.10	Evaluation of the Standard Deviation-based Method .	77
4.11	Kurtosis-based Method	79
4.11.1	Long-term Test	80
4.11.2	Short-term Test	81
4.11.3	Message Rate	82
4.11.4	Algorithm	82
4.12	Evaluation of the Kurtosis-based Method	82
4.12.1	Detection Performance	83
4.12.2	Spatial Dependence	85
4.12.3	Multiple Receivers	86
4.12.4	Message Rate	86
4.12.5	Detection Distance	89
4.12.6	Comparison With Other Systems	90
4.13	Limitations	92
4.14	Conclusion	94

4.1 Introduction

In order to begin approaching the topic of ubiquitous sensing for security in smart homes, thus addressing our first research question (**RQ1**), this chapter investigates how devices that are already deployed in smart homes can be leveraged to provide a security service for which they are not purpose-built. To this end, we propose a system that relies only on readily available radio signal strength information to detect drone-based privacy-invasion attacks—while requiring nothing but a software update. Furthermore, we study to what extent the placement of the devices providing these signal strength measurements affect their detection performance. This detection performance determines the devices’ ability to perform their additional security function, and its analysis thus tackles **RQ2**.

We choose drone detection as our first system based on ubiquitous sensing due to increasing worries by home owners about privacy violations perpetrated by drones, and because of the wide-spread use of wireless devices capable of providing the radio signal strength information that is required for our proposed detection system.

Drones have become enormously popular in recent years. Since the release of the Parrot AR.Drone in 2010, examples of the technology have exploded into the public consciousness. The use of drones—otherwise known as unmanned aerial vehicles (UAVs)—is no longer restricted to military and professional domains. It has opened up to the layperson as well. Self-stabilizing multirotor designs that automatically hover upright have made piloting drones easily accessible for newcomers. Meanwhile, advances in sensor design and image processing have brought sophisticated flight-assist features to further ease the burden. Modern consumer drones can be flown using a handheld controller and a smartphone, and while safety records do vary wildly it is completely achievable for a novice to operate this type of aircraft without incident despite no prior training whatsoever.

The opportunity to take to the skies without arduous preparation has clearly appealed to many people. Industry analyst firms have reported sales of drones increasing every year since 2014, with the global drone market expected to grow to \$63.6 billion by 2025 [186]. The US Federal Aviation Administration reported in January 2018 that there were over a million drones on their national register [187].

Sales by the market-dominant DJI Technology were reported at 375,000 units in 2017, with the majority intended for photography [188]. Indeed, photography is one of the most popular uses for drones and the inclusion of high-quality, onboard camera equipment is a recognized selling point for a device.

However, the rise of consumer drones has brought many problems with it. Political figures have been targeted in drone-based protests [189, 190], the repeated appearance of a drone closed Gatwick airport for two days in December 2018 [191], and drones have even been used to deliver weapons, drugs and other contraband into prisons [192]. More localized abuses are often reported as well and there has been an increasing unease in the general population about privacy invasion by drones carrying high-fidelity camera equipment. As most consumer models are outfitted with cameras for live-view video during flight, drones can fly over fences to see into nearby gardens or even look into windows to observe the interior. In Seattle, a woman called the police after spotting a drone flying in front of her

26th-floor apartment window, observing her partially-undressed inside [193]. In a famous landmark case, a man from Kentucky was arrested after shooting down a drone over his property. He explained this act by saying that the drone was spying on his sunbathing daughter [194].

While systems for detecting drones are available, they are specialized and costly, intended for law enforcement use to protect secure sites and VIPs, rather than for normal citizens. By contrast, we focus on these threats to private individuals. An attacker can launch a drone from a distant location and fly it to an otherwise-inaccessible private area, such as outside a top-floor window within a walled garden, and freely observe the occupants. To launch such an attack the attacker has to move close enough to the window to get a full view of the inside of the house. This view of the interior depends on the field-of-view (FOV) of the window, as Figure 4.1 demonstrates. As a window is a distant aperture of fixed size, the FOV initially increases very slowly during the approach before expanding rapidly once the drone is at close range. The occupants require a system that alerts them to the presence of an impending privacy invasion, with sufficient warning to allow them to react accordingly and before the drone gets close enough to allow observation. The system must be easy to deploy without specialist knowledge and able to detect a wide variety of drones based on intrinsic properties of UAV activity—both to be robust to the rapid pace of advancement in drone technology and to avoid spurious alarms. To be most readily-applicable, the system should avoid additional hardware requirements and take advantage of existing sensing capabilities of modern smart home systems.

In this chapter, we present a system that detects the presence of a live-streaming UAV that is being used to invade someone’s privacy in their home. We make use of readily-available commercial, off-the-shelf (COTS) hardware to measure variations in the strength of the drone’s radio control signal and develop metrics that identify signal properties inherent to drone flight. By using a radio-based approach, the system is equally effective in light or dark conditions and as the signal measurements are commonly provided by radio chipsets, the system also has the potential to be



(a) View into window at distance of about 10m. The FOV is narrow and the reflections mask view. Little of the room is visible.



(b) View into window at close-range of roughly 2m. The FOV is wider and the reflections less dominant. The door, curtains, red walls and bed can be seen.

Figure 4.1: Examples of effect of distance on surveillance footage quality.

a software-only change to existing smart home devices. In developing the system, we make the following specific contributions:

- We develop a model of UAV-based privacy-invasion attacks.
- We derive statistical metrics to identify drone movement and proximity. These metrics can be derived from signal measurements provided by typical in-home devices (**RQ1**).
- We propose mechanisms to overcome attempts by an attacker to avoid detection, such as varying their speed or flight pattern.
- We evaluate the system in a real-world scenario using popular consumer drones and various moving benign transmitters.
- We demonstrate that our detection method can make use of receivers in non-ideal locations and that it can leverage multiple receivers at once to improve detection performance (**RQ2**).

4.2 Background

The market for consumer drones is contested by many different manufacturers, but at time of writing it is dominated by only a few companies, namely DJI Innovations (75% market share), Yuneec (5%), 3D Robotics (3%) and Parrot (2%) [195].

The majority of drones provide a live video stream back to the operator—so-called “first-person view” (FPV). In all but the simplest of flights it is a great benefit to the pilot to be able to see from the perspective of the drone itself. The user can commonly connect to the drone with their smartphone, tablet or a dedicated controller to receive the video, and often record the footage in higher resolution as well. Interoperability with existing user devices has played a great role in enhancing the appeal of these consumer UAVs. While some higher-end models use proprietary communication technologies for their video downlink, most drones rely on common 2.4 GHz or 5.8 GHz Wi-Fi instead. Some even use Wi-Fi for the control channel of the drone. In these cases, the drone can be controlled completely by an app. Otherwise, a separate, dedicated controller is needed and the Wi-Fi connection is

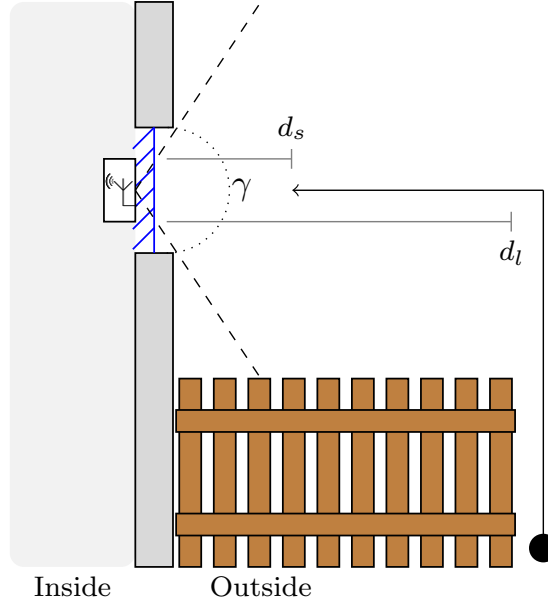


Figure 4.2: The attacker launches the drone at launch distance d_l and flies it to surveillance distance d_s to carry out the attack. The detector is installed in the window. The FOV γ of the window limits the area that is in LOS of the detector.

only used for the live-view video. While we focus on Wi-Fi throughout this study, the approach could be applied effectively for any receivable RF technology as it depends solely on measurement of received power.

Airspace regulations differ between jurisdictions in terms of the requirements placed upon drone operators. Almost all mandate that minimum separation distances are observed between the drone and any surrounding persons or property. In the United Kingdom, the Civil Aviation Authority (CAA) classes drones equipped with cameras as “small unmanned surveillance aircraft” and requires that they [196]:

- fly no closer than 50m from any persons, buildings or vehicles that are not under the control of the operator
- stay below a 400ft flight ceiling
- stay within line-of-sight of the operator

4.3 Adversary Model

The attacker uses a radio-controlled drone, with an onboard camera and a live video feed, to invade the privacy of individuals inside a building. Specifically, the attacker uses a drone to look into a window that they would otherwise have a poor view of due to distance and/or height of it. Whether they are motivated by voyeuristic pleasure, a desire to observe private information or even simple curiosity, high-quality camera footage is the means to satisfy their aim. It is also necessary to stream live footage back to the operator throughout, in order for them to pilot the drone accurately¹.

The attacker does not modify their drone to help achieve their goal, although they can fly it in any manner that is required. This is consistent with an opportunist who acts inappropriately with their equipment but lacks the skill, resources or premeditated will to optimize for an attack.

This type of attacker is akin to a zero-effort attacker. Stronger types of attackers might use their awareness of the system to either time an attack based on currently perceived sensor data (*e.g.*, when the radio environment is particularly noisy), or manipulate the signal that the sensors can receive (*e.g.*, by varying the signal power or by hiding the output signal entirely). While these types of advanced attackers are out of scope for this chapter, we include a discussion of how they might affect the proposed drone detection system in Section 4.13. Additionally, we show the impact of similarly powerful attackers on a system based on ubiquitous sensing in Chapter 5.

We model the attack as follows—visualized in Figure 4.2. The attacker does not have access to the premises and is thus positioned some launch distance d_l away from the target window. The launch distance depends on the size of the property and the signal range of the drone controller. They fly the drone from their starting position towards the target window until it can establish line-of-sight (LOS) into the

¹In our experience during data collection, this requirement was surprisingly stringent. Even at small distances it quickly becomes difficult to estimate the relative positions of the drone and obstacles in 3D space from only a single viewpoint. Most relevantly, we found it particularly challenging to know whether a drone had a clear shot of a window without watching its ‘first-person view’ (FPV) feed. Indeed, not paying due attention to this feed and flying based on the pilot’s view alone quickly led to the most serious crash we encountered, wherein the drone contacted the building wall and fell 9 feet to ground, immobilized.

room and is close enough, at surveillance distance d_s , to observe a significant portion of the interior at the required quality. Figure 4.1 compares the drone's viewpoint at different distances. The quality of the footage is determined not only by the distance, but also the camera resolution and the field-of-view that the window allows, taken as γ . The speed of the approach is limited only by the drone's capabilities and so its value v_{max} represents an upper bound on the speed of contemporary drones.

An attack is divided into three distinct phases:

Approach The drone is launched and flies from the start location towards the target window via some flight pattern. The flight pattern may be dictated by obstacles between the attacker and the target, or at the attacker's discretion in an attempt to avoid detection. At some point during the approach, line-of-sight must be established in order to permit surveillance and the drone must get close enough to have a detailed view inside the building.

Surveillance The drone establishes a good surveillance position, it hovers for a period of time and records video of the building interior. The movement of the drone in this phase is kept minimal in order to increase the quality of the recorded footage.

Escape Once the surveillance has been conducted, the drone is piloted away from the window and returns to a collection site.

While the attacker cannot alter the fundamental operation of their drone, they can vary the flight pattern and speed in an attempt to avoid detection. For example, they may try to approach as quickly as possible or as slowly as possible. They may attempt to mimic hobbyist flight or stay out of sight of the window for as long as possible during approach. The attacker is limited only by the bounds of the drone's maneuverability.

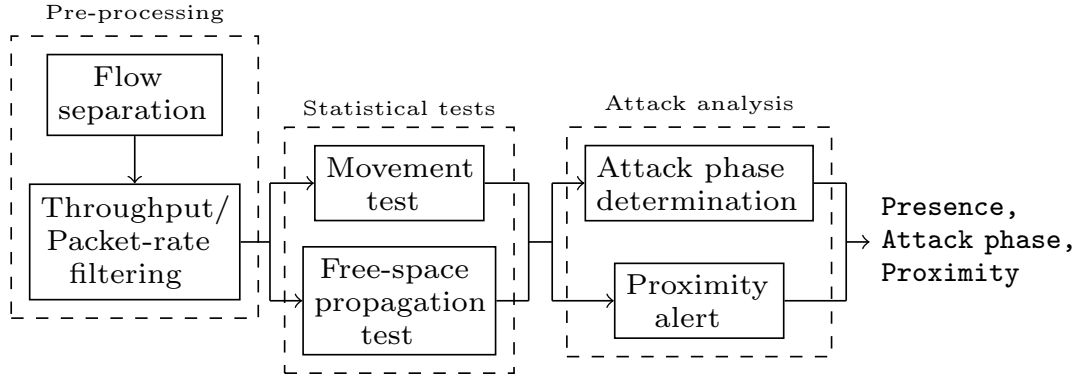


Figure 4.3: Flow diagram of the detection algorithm.

4.4 System Model

A radio receiver is mounted in the building interior, configured to passively listen for all available traffic and record the received signal strength (RSS). This is generally-applicable to any radio technology, but the most straightforward example is a Wi-Fi receiver operating in monitor mode and hopping channels to cover the entire band. We consider that implementation throughout this chapter. While an ideal receiver position is at the target window itself, as in Figure 4.2, the system does not constrain the receiver position; it need not be located in the same room as the target window, nor even within line-of-sight.

With an input stream of traffic and associated RSS values, the system then proceeds in three broad phases, as shown in Figure 4.3. In the *pre-processing* phase, the received traffic is separated into flows and filtered based on throughput. During the *statistical tests* phase, the system computes live statistics for each flow, derived from the RSS measurements, over a set of rolling time windows of various sizes. It also monitors the baseline statistics of packets from a known, static transmitter (such as the user’s Wi-Fi access point). Analysis is performed on the statistical metrics across multiple time windows, along with knowledge of the baseline, to establish whether individual transmitters are moving and operating in free space, consistent with drone behaviour. If the statistical tests pass then the system decides that the moving transmitter is indeed a nearby drone. Subsequently, the *attack analysis* phase determines the attack phase that is in progress and whether the

drone has reached a pre-defined detection distance d_d in close proximity. We expand on each stage below and present specific implementations in Section 4.5 and 4.11.

4.4.1 Pre-processing

The first step towards detection is the separation of different data flows. Given the assumption that the drone is unmodified and that communication makes use of a standard protocol (*e.g.*, the IEEE 802.11 Wi-Fi standard), flows are separated by MAC address. Note, that if the MAC address range of a drone vendor is known, a MAC filter can be a helpful addition to the system. However a MAC filter is not sufficient to detect a privacy invasion attack as a thus detected device could simply be an innocuously operated drone nearby. Instead, our system detects the approach of a drone as well as its proximity to the protected site. Furthermore, as new models of drones are released regularly and because new vendors frequently enter the market, a MAC-based approach would have to be constantly updated to ensure that all drones can be detected. Our system therefore does not rely on a MAC filter based on vendor address ranges, but can be easily extended to incorporate one. The provision of a continuous, live video stream from the drone to the controller, necessitates a consistent, high bandwidth utilization on the communication channel². The detection process excludes flows that do not display this characteristic. Note that this is not a sufficient condition for drone video streams and thus only used as a pre-filtering technique to avoid unnecessary computations. To distinguish drones from other moving or stationary transmitters, the statistical tests described in the following subsection are used.

4.4.2 Statistical Tests

An airborne drone operates largely in free space, and must establish LOS to the window (and thus the detector) in order to conduct a privacy invasion attack. As such the transmission environment is uncluttered and the received signal strength can be expected to be dominated by the direct-path component. We neglect

²For example, a Parrot Bebop drone creates 400 packets/s, for a throughput of 500KB/s, when providing 720p video at 30 frames-per-second.

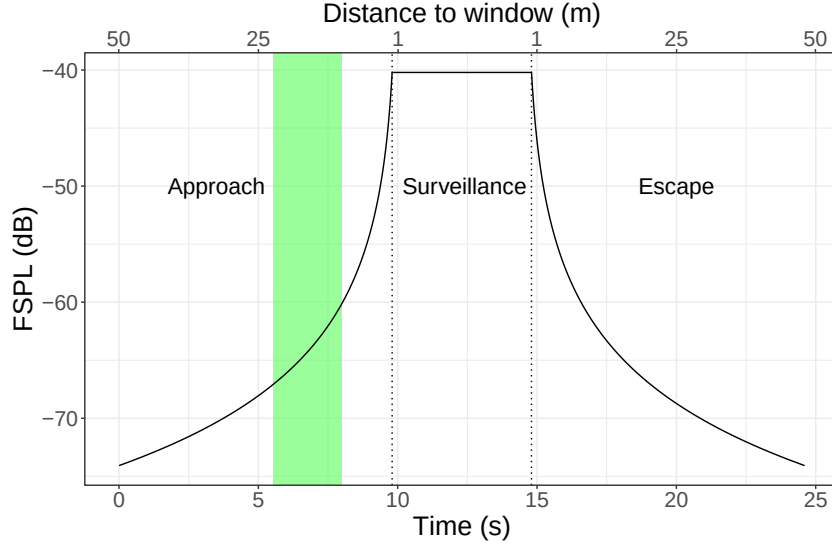


Figure 4.4: Simulation of the Free Space Path Loss during a privacy invasion attack with 5s surveillance. The speed of the drone is $5m/s$, the surveillance distance d_s is $1m$, the green shaded area shows the time window during the approach when the drone can get detected if a detection distance d_d of $10m$ is chosen. The detection distance is the distance at which the system is configured to trigger a drone proximity alert.

multipath effects due to nearby houses and the ground as the drone has this strong, short-range LOS connection. These free-space propagation conditions yield the RSS behavior shown in Figure 4.4—illustrating the path loss changes over time for an attack with a shortest-path approach and constant speed.

The system applies statistical tests to a flow to establish whether it is likely to represent a drone or not. The tests are based upon a comparison of the statistical tests when applied over short and long rolling time windows. Intuitively, the concept is as follows: the message rate of live-streamed video is sufficiently high that a drone will move very little between individual messages, but the drone’s movement is the dominant factor in changing the signal strength over a longer period. Short-term tests can therefore be used to determine the propagation environment between the transmitter and the receiver, while long-term tests can establish whether a transmitter is moving.

Combining information from both time periods allows the system to identify transmitters that are both operating in LOS and are moving—which are likely to be drones. Other moving transmitters on the ground or indoors can be expected

to have more short-term variation due to changing multipath effects. Whereas immobile transmitters in a static environment are expected to have a stable long-term standard deviation and those in a dynamic environment are expected to experience substantial short-term changes.

4.4.3 Attack Analysis

All flows considered to be drones are examined to establish whether or not they are being used to mount a privacy invasion attack. The approach behavior relative to the receiver is determined by monitoring the long-term RSS trend: whether it is increasing, stable or decreasing. A proximity test is then applied to drone flows that appear to be approaching or hovering, to determine whether the drone has reached d_d and is about to begin surveillance.

4.5 Standard Deviation-based Method

A suitable implementation of the system model's statistical tests and attack analysis can be derived based on the standard deviation of the RSS values for a given transmitter. We describe that derivation here.

To establish a baseline, we define a noise threshold σ that represents the standard deviation of the noise the receiver is subject to. This is determined in an automatic calibration phase by measuring the noise from a known transmitter in a static environment (*e.g.*, Wi-Fi access point).

Given the high packet rate exhibited by a live-streaming drone, in a sufficiently short period the drone does not move enough for the change in RSS due to movement to be distinct from noise. An immobile, LOS transmitter in a static environment exhibits changes in signal strength that are primarily due to measurement noise and cross-traffic interference. Hence, we expect a standard deviation that is close to the general noise level σ , in such a timeframe. Over a long time period, however, the RSS from a moving drone is mainly affected by that movement. So, whilst the standard deviation alone may not be enough to confirm LOS transmission [197], we

can additionally use the long-term changes that are detectable by having a higher standard deviation than for immobile transmitters [198].

In order to apply both tests, the system must first determine appropriate short and long window sizes. To find a window size where the change in signal strength stays below the noise level, we consider the part of the approach with the maximal possible change. This change occurs when the drone travels the final stretch at maximal speed before it arrives at surveillance distance d_s ³.

We assume that the drone transmits with packet rate r , and d_s , $v_{\max}$ and transmission frequency f are given. At packet rate r we receive $N = w \cdot r$ transmissions in a time window of length w . As a simplification, we use the free-space path loss (FSPL), or the amount of energy that the signal would lose in a free-space environment, to represent the received signal strength of the drone. Then the free-space path loss of each of these N measurements is given by [199]:

$$x_i = 20 \log_{10} \left(d_s + \frac{i}{r} \cdot v_{\max} \right) + 20 \log_{10}(f) - 27.55$$

The unbiased sample standard deviation can be computed with:

$$s(N) = \sqrt{\frac{1}{N-1} \sum_{j=1}^N x_j - \bar{x}}$$

where $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$ is the sample mean.

We then have to compute the maximal window size for which the standard deviation is below the noise threshold σ :

$$w_s = \max \{w | s(w \cdot r) < \sigma\}$$

Then σ bounds the standard deviation of the random variable $FSPL$ within window w_s .

We consider the signal strength as the sum of random variables $FSPL$ for the free-space path loss and X_N for the noise. Then the variance of the sum of

³This is unlikely to happen in reality, as it is too hard to control a UAV at that speed so close to the window. Nevertheless, it helps to determine the largest change that is physically possible and caused by movement alone.

these random variables is given by [200]:

$$\text{Var}(FSPL + X_N) = \text{Var}(FSPL) + \text{Var}(X_N) + 2\text{Cov}(FSPL, X_N)$$

We know from above that $\text{Var}(FSPL) < \sigma^2$ and $\text{Var}(X_N) = \sigma^2$. Additionally, we know that $FSPL$ and X_N are uncorrelated, as they are independent random variables. This means that $\text{Cov}(FSPL, X_N) = 0$ and it follows that:

$$\text{Var}(FSPL + X_N) < \sigma^2 + \sigma^2 + 2 \cdot 0 = 2\sigma^2$$

Consequently, the short-term *free-space propagation* test fails when the standard deviation of the measured samples during w_s surpasses $\sqrt{2}\sigma$.

Analogously, we compute the larger window size to detect movement by choosing an expected minimal velocity v and by looking at the first stretch after launching at distance d_l . In this case, the free-space path loss for the N corresponding measurements is:

$$x_i = 20 \log_{10} \left(d_l - \frac{i-1}{r} \cdot v \right) + 20 \log_{10}(f) - 27.55$$

This time, we are looking for the minimal window size for which the sample deviation is above the threshold:

$$w_l = \min \{w | s(w \cdot r) > \sigma\}$$

Again, using the independence of $FSPL$ and X_N it follows that:

$$\text{Var}(FSPL + X_N) = \text{Var}(FSPL) + \text{Var}(X_N) > 2\sigma^2$$

As a result, the *movement* test is successful if the measured samples during w_l have a standard deviation that is higher than $\sqrt{2}\sigma$.

4.5.1 Attack Analysis

The attack phase can be deduced by taking the difference between the mean of the first and the second half of w_l : $\Delta x = \bar{x}_{[1, \lfloor \frac{N}{2} \rfloor]} - \bar{x}_{[\lceil \frac{N}{2} \rceil, N]}$. The sign of Δx determines the current attack phase: if the drone is approaching, Δx is larger than zero, whereas

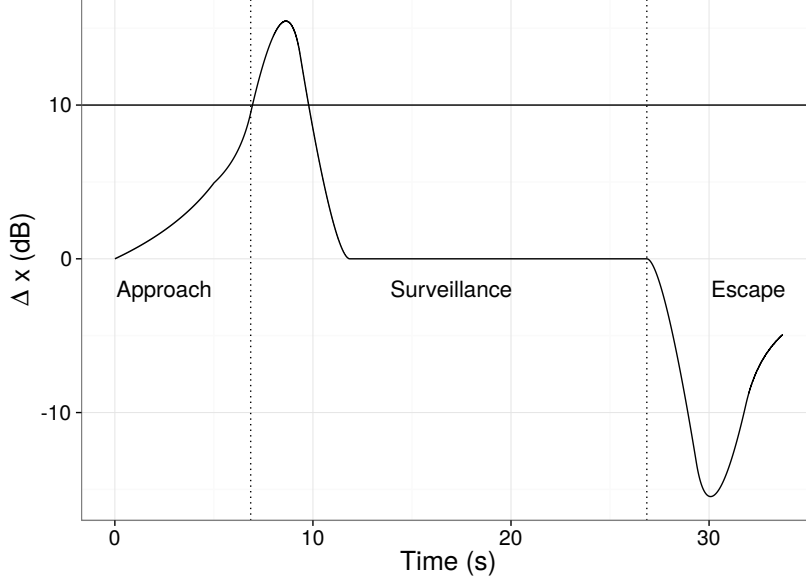


Figure 4.5: Attack phase and proximity detection for a simulation with $v = 7\text{m/s}$, $d_l = 50\text{m}$, Gaussian noise with $\sigma = 2\text{dB}$ and $\sigma_p = 10$. The black horizontal line is the proximity threshold for $d_d = d_s = 1\text{m}$. We used $w_l = 5\text{s}$ as determined in Figure 4.9.

values lower than zero indicate that it is escaping. If the sign of Δx is zero, the drone is not moving. However, this is not sufficient to decide whether the surveillance phase has started. It could just be hovering at a larger distance from the window, without being able to observe the interior. Therefore, we have to take the *proximity* to the window into account. We assume that the approach has been detected previously. Let w_l be the window size in which the detection happened and let v be the corresponding drone speed. Then the drone has arrived at detection distance d_d , if Δx as defined above is larger than or equal to the following threshold σ_p :

$$x_i = 20 \log_{10} \left(d_d + \frac{i}{r} \cdot v \right) + 20 \log_{10}(f) - 27.55$$

$$N = w_l \cdot v, \Delta x \geq \sigma_p$$

To detect the start of the surveillance phase, d_d can be set to d_s . One can see in Figure 4.5 that the time window has to catch up to detect the surveillance phase. Therefore, it is important that an alarm is already raised as soon as the proximity threshold is breached, rather than waiting for Δx to reach zero, to ensure a timely detection. This is especially true, if the approach was very slow as w_l might cover the whole surveillance period otherwise.

Table 4.1: Example approach flight patterns

Approach pattern	Description	LOS
Straight	Drone follows shortest path to window	Constant
Zig-zag	Drone follows zig-zag pattern, resulting in variable effective approach speed	Constant
Back and forth	Drone approaches, backtracks, then progresses again, resulting in an effective approach speed that is sometimes negative	Constant
NLOS	Drone avoids LOS until shortly before surveillance phase	Emerges

4.5.2 Detection Range

The detection range d_r is the maximal possible detection distance d_d . This is the closest distance to the window that the drone can reach from d_l within w_l seconds. To compute the time until our system can detect an attacker, we take w_l as computed above: $t_d = w_l$. Then the detection range d_r is: $d_r = d_l - t_d \cdot v$. The detection range depends on d_l , whereas the movement speed v has little effect on it, and only shows a slight variation due to the sampling rate. As we do not know the actual speed of the attacking drone in advance, we have to compute a range of different window sizes in parallel to guarantee a timely detection.

4.5.3 Approach Patterns

The case studied above corresponds to a straight approach to the window. As noted in Section 4.3, the attacker can vary their approach in an attempt to avoid detection. We consider a set of example flight patterns; detailed in Table 4.1 and visualized in Figure 4.6. Whilst these patterns are not exhaustive, they demonstrate some extremes of the behavior of an attacker. The first three patterns have in common that the attacker is in LOS of the window for most of the attack; the main difference between these three patterns is the effective speed of the drone or the speed with which the drone closes the distance towards the window as opposed to its actual movement speed. The fourth pattern, the non-line-of-sight (NLOS) approach, ensures that the drone is only in LOS of the window for a short period

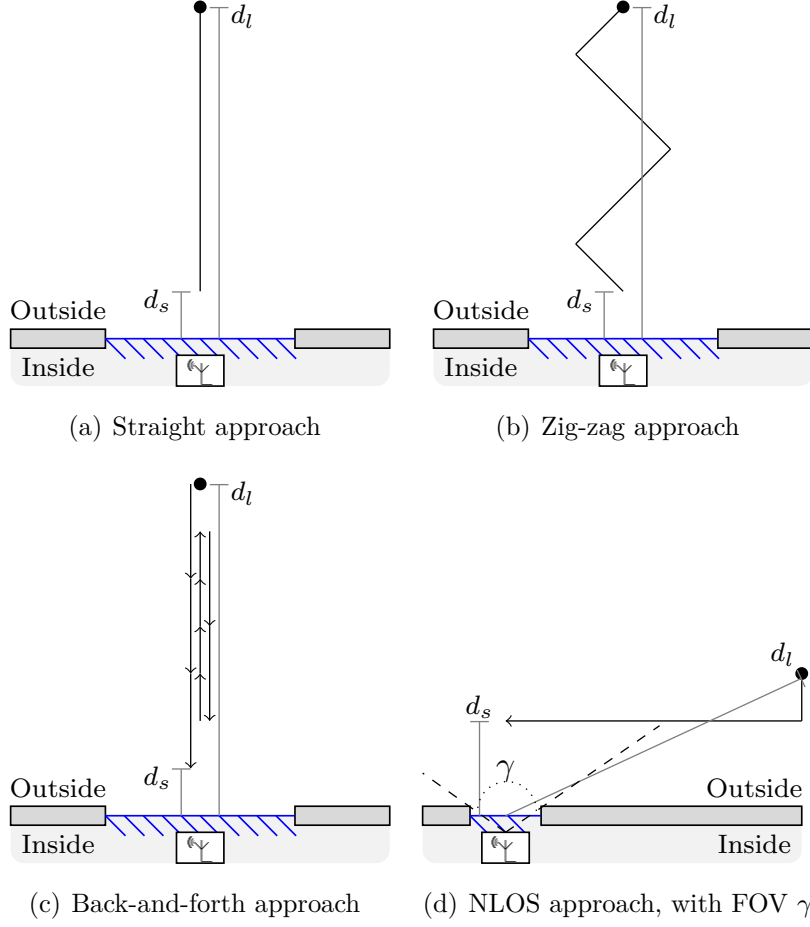


Figure 4.6: Various example approach patterns

of time. By approaching from the sides, above or below like this, the attacker can give any detection approach, whether human or machine, less opportunity to detect the drone prior to surveillance beginning.

In contrast to the straight approach, flying towards the window using a zig-zag approach results in a lower effective velocity towards the window, which in turn affects the choice of appropriate w_l for the movement test. We simulated the standard deviation for a zig-zag approach with two different sliding windows; one for the actual speed v and another for the effective velocity v_{eff} . In Figure 4.7, one can see that the detection with a window size that is too short for the effective velocity causes a delayed detection. Additionally, the shorter window size captures the zig-zag motion of the approach, whereas these erratic changes get smoothed out by the larger window size of v_{eff} . A similar effect is caused by a back-and-forth

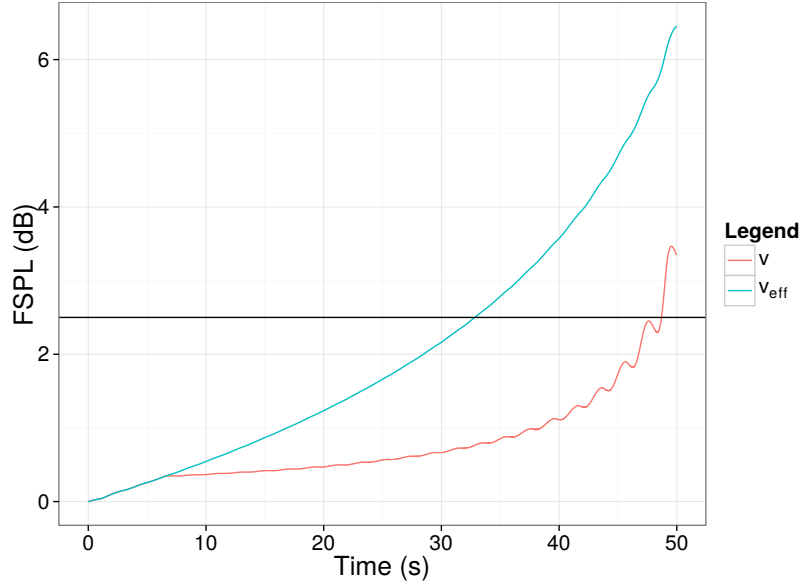


Figure 4.7: Simulation of a zig-zag approach with two different choices of movement test window w_l . One is optimized for the actual speed, the other one for the effective velocity. The latter allows a faster detection.

approach. The choice of w_s for the free-space propagation test, on the other hand, is unaffected by the approach pattern.

If most of the approach is done out of LOS of the receiver, our assumptions about free-space propagation do not hold for the larger part of the approach. Therefore, the part of the approach that can be used for detection with our method gets reduced significantly. This distance depends on the surveillance distance d_s and the FOV γ of the window. It is therefore possible that detection might fail under extreme conditions. Depending on the FOV γ , noise threshold σ and surveillance distance d_s , the approach in LOS can be too short to result in a notable change of the signal strength. While this is a theoretical possibility, our practical testing suggests it is a remote one. Refer to Figure 4.8 to see how the detectability of a NLOS approach depends on γ and d_s .

4.5.4 Detection Algorithm

In order to detect drones, we compute the sliding windows for the standard deviation of the signal strength using w_s and increasing values of w_l . We proceed like this

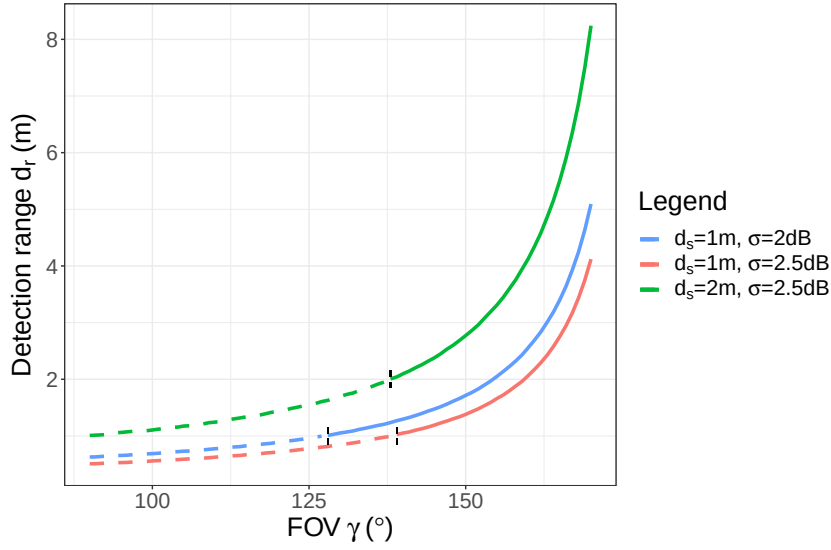


Figure 4.8: Detection ranges for the NLOS approach. If the detection range d_r is smaller than surveillance distance d_s , the detection will fail.

until we find a value for which the standard deviation is below the threshold within w_s but breaks the threshold for w_l .

We then use w_l to determine the attack phase by computing Δx . For values greater than zero the drone is approaching. If Δx additionally exceeds σ_p , then the drone has reached d_d and an alarm is raised. If d_d is set to d_s then the rolling windows should be reset after a proximity alert to make sure that the surveillance phase is properly detected. The surveillance alarm lasts until the start of the escape phase is detected.

Pseudo-code for the detection algorithm can be found in Algorithm 1, supporting the overall flow diagram in Figure 4.3. See Figure 4.9 for the movement and free-space propagation tests of the detection algorithm. This figure shows the rolling standard deviation for a moving transmitter and a receiver that is exposed to Gaussian noise. The attack phase tracking and proximity alert for the same simulation are pictured in Figure 4.5.

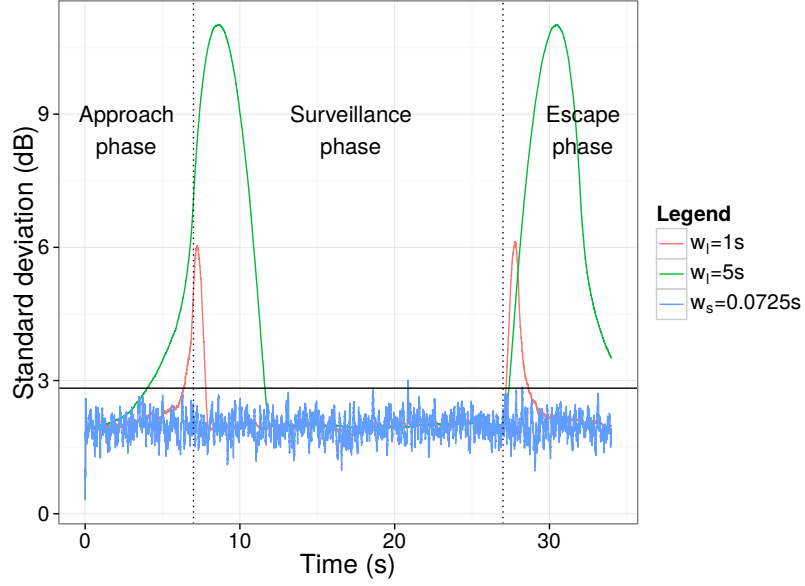


Figure 4.9: Detection algorithm for a simulation with $v = 7m/s$, $d_l = 50m$ and Gaussian noise with $\sigma = 2dB$. It computes the standard deviation for the *free-space propagation* test during w_s and several window sizes w_l in parallel for the *movement* test. The black horizontal line is the noise threshold. It is first broken by $w_l = 5s$, making this the preferred window size.

Algorithm 1: Standard deviation-based detection algorithm. Here, s , $phase$ and $proximity$ are the tests introduced in Section 4.5. A full description for the algorithm is given in Section 4.5.4.

Data: Data flow of one transmitter

Result: Boolean: Drone alarm, String: Attack phase

begin

```

    Drone alarm, Attack phase  $\leftarrow \emptyset$ 
    for Increasing  $w_l$  until  $v_{\min}$  do
        if  $s(w_s) < \sqrt{2}\sigma$  and  $s(w_l) > \sqrt{2}\sigma$  then
            if  $phase(w_l) > 0$  then
                Attack phase  $\leftarrow$  Approach
                if  $proximity(w_l)$  then
                    Drone alarm  $\leftarrow$  True
                    if  $d_d == d_s$  then
                        Attack phase  $\leftarrow$  Surveillance
                         $reset(w_l)$ ;
            else if  $phase(w_l) > 0$  then
                Attack phase  $\leftarrow$  Escape

```

4.6 Preliminary Experiment

To demonstrate construction of our detection system using COTS hardware, we made use of a Raspberry Pi Model A⁴ unit as a receiver. As this model lacks on-board Wi-Fi hardware, a Wi-Pi USB Wi-Fi adaptor was employed to receive packets. The Raspberry Pi ran the dumpcap utility⁵ to capture Wi-Fi traffic, with the Wi-Pi adaptor in monitor mode to allow traffic capture without having to associate to any network.

The preliminary real-world experiment took place at a secluded⁶ domestic property in Devon, United Kingdom. The property was an old farmhouse with thick, stone walls that were expected to attenuate Wi-Fi signals heavily. The property was surrounded by open land, allowing a variety of approach flight paths.



Figure 4.10: Deployment of the receiver

The Raspberry Pi was installed in the window of a first-floor bedroom at the back side of the house. Figure 4.10 shows the deployment location and the receiver *in situ*. The Raspberry Pi was connected to a controlling laptop via a Powerline network to avoid introducing cross-traffic on the wireless channel. For calibration, we used the beacons of an accesspoint that was located in the kitchen on the ground-floor to gather enough calibration packets in a timeframe of thirty seconds.

⁴www.raspberrypi.org

⁵Part of the Wireshark network protocol analyzer (www.wireshark.org)

⁶To comply with CAA regulations as detailed in Section 4.2 and avoid disruption to others. The nearest property that was not under our control was over 350m away.

The signal strength measurements of these packets were then used to compute the standard deviation σ of the baseline noise, as described in Section 4.5.

Two very popular drone consumer drone models were used to mount our simulated privacy attacks. The primary candidate was a DJI Phantom 3 Standard, with the experiments being repeated using a Parrot Bebop unit for comparison. Both drone models were operated on the 2.4GHz Wi-Fi band. The two drone models are pictured in Figure 4.11. Unfortunately, some of the experiments were cut short due to time restrictions and the English weather; especially those from the second series using the Parrot Bebop. For this preliminary experiment, we did not yet include cross-traffic by static or mobile ground-based transmitters. A full experiment enabling an investigation into how false alarms affect the system can be found in Section 4.9.



(a) DJI Phantom 3 Standard



(b) Parrot Bebop

Figure 4.11: Drones used in the real-world experiments

We performed attack runs for each of the approaches described in Section 4.5; in each case we used the first floor window that the Raspberry Pi receiver was in as the privacy-invasion target. The NLOS approach was further split into approaches over the roof and around the house to include both a descent from above and reaching the window from the side. Every approach in the series was performed with three run repetitions. The launch distance was between 55m and 65m for the LOS approaches. For the NLOS approaches the launch distance was much shorter

Table 4.2: DJI Phantom 3 experiment series.

Approach pattern	Distance (m)				Max. Speed (m/s)			
	Run	1	2	3	Run	1	2	3
Straight		57	61	61		5.94	6.24	6.35
Zig-zag		61	62	62		6.36	6.66	4.72
Back-and-forth		62	61	61		4.24	4.43	5.38
NLOS over the roof		28	30	30		2.78	3.92	3.62
NLOS around the house		35	31	33		2.48	3.48	4.06

at around 30m, as they started from the front side of the house, with the operator moving to keep the drone in view during the approach. No run exceeded a top speed of 7m/s (approx. 25kph/15mph), as it is hard to control the drone at high speeds close to a building. The GPS flight data for each series were recorded by the Phantom. Some example traces are shown in Google Earth⁷ plots in Figure 4.12. One can see that the GPS data became less precise close the window, as the drone no longer had a clear view of the GPS satellites. Specific details of the Phantom experiment series, as extracted from the GPS flight data, can be found in Table 4.2.

4.7 Preliminary Evaluation

In this section, we evaluate how well the metrics from Section 4.5 are able to capture drone behavior using the data from our preliminary experiments. This preliminary evaluation does not include a false alarm analysis, which will be provided in Section 4.10. We use only the data from the Phantom for the analysis of the movement test, attack-phase tracking and proximity detection. The results for these are similar for both Phantom and Bebop and, as described in Section 4.6, we have a more complete set of runs for the Phantom. However due to a technical issue, only the control channel data was captured for the Phantom, so the packet rate is quite low at 40 packets per second. While this is not a problem for the large-window tests, it leaves only about four packets within w_s . Hence, we used the data from the Bebop series to examine the free-space propagation test.

⁷earth.google.com

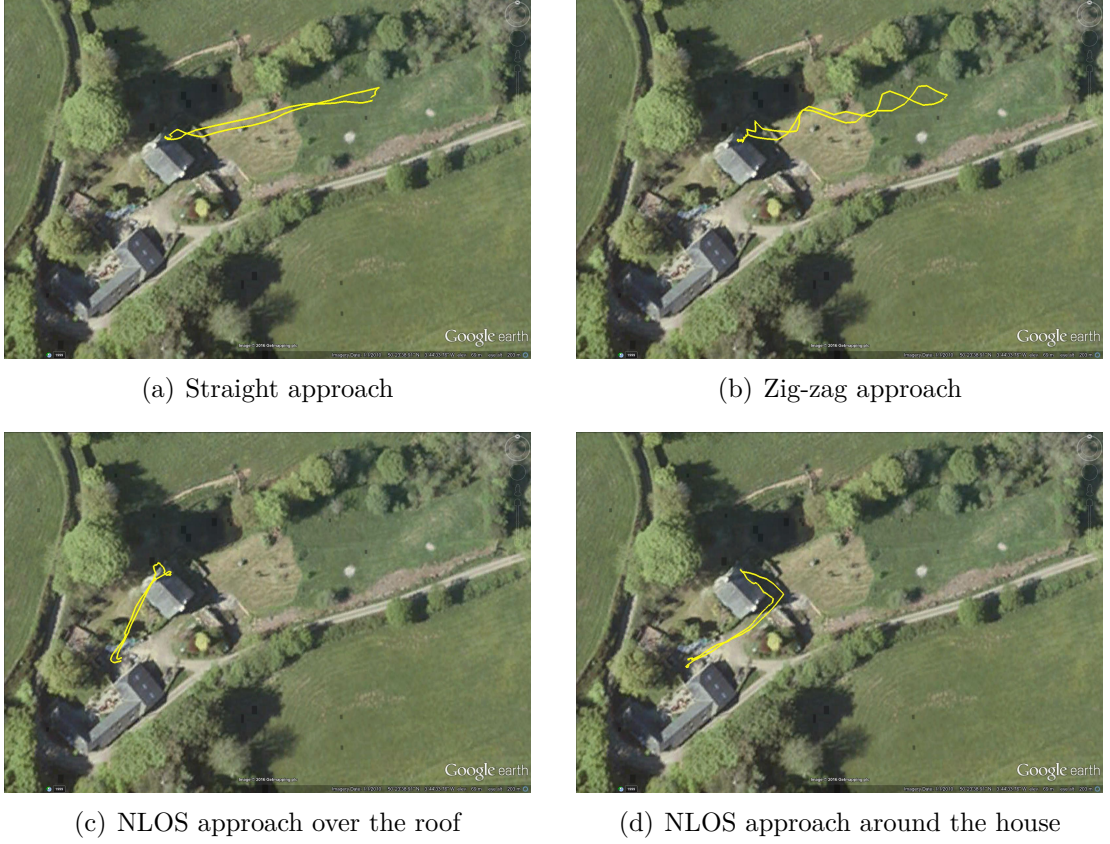


Figure 4.12: Different flight patterns during the approach

In our evaluation, we selected system parameters as given in Table 4.3. Recall from Section 4.5 that longer window sizes w_l are necessary to capture slower approaches whereas shorter ones will trigger earlier alerts for faster drones. The system therefore runs with several different window sizes concurrently. The minimal detection range using these parameters was 24.5m.

4.7.1 Straight Approach

Examining movement-tracking first, in Figure 4.13(a) three windows of rolling standard deviation are displayed for the first run of the Straight Phantom set. The drone launched at (1) at a distance of 57m from the window; the first increase is due to the initial ascent, but is not enough to break the threshold. There is a short spike in the 5s window before the actual detection happens which only lasts for a few samples. At (2) the drone started accelerating towards the window. Due to

Table 4.3: System parameter selections used in Evaluation

Parameter	Value	Description
d_s	1m	We stayed within one to two meters of the window during the surveillance phase
d_l	50m	All the LOS approaches started further than that
w_s	0.1s	Corresponding to a maximum velocity of 10 m/s
w_l	5s, 10s, 15s, 30s	Corresponding to the following respective minimal velocities: 5.075, 2.535, 1.69, 0.845 m/s
σ	1.75	Derived according to Section 4.5; hence the noise threshold is $\sqrt{2} \cdot 1.75$
σ_p	10	The threshold for the proximity alert; derived from d_s

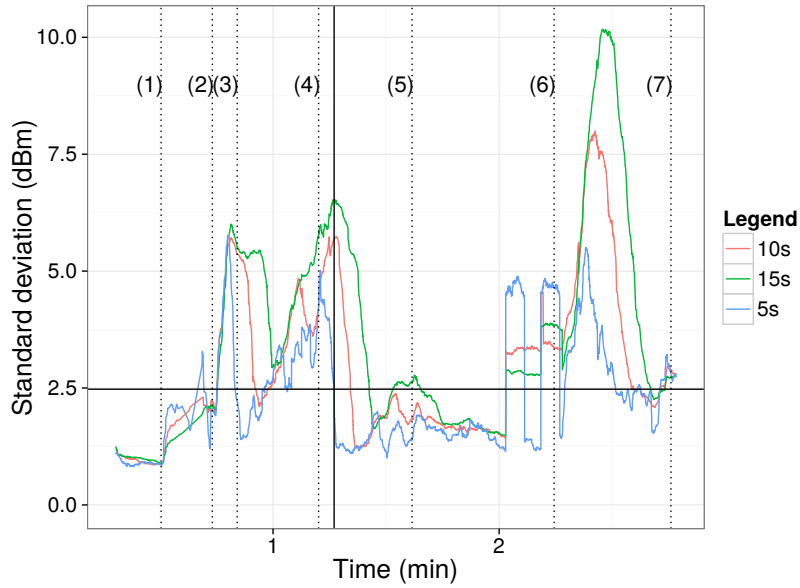
its slow speed at the time, the 10s window is the first to detect it, albeit by a very small margin. The detection happens surprisingly fast after only one and half a meters traveled since takeoff. From the model in Section 4.5 we had expected the drone to be on target height and approach speed when we received our first packet, however the detection happened much earlier than expected because the takeoff and the varying speed due to acceleration increase the measured standard deviation. In comparison, if we were to start detection during the second acceleration period starting at (3) at a distance of 51m, the drone would be detected 37m from the window. This is closer to the expected value of 25.15m for this distance but still earlier than predicted. This can be attributed to the drone not traveling as close to a constant speed as expected. The velocity decreased at (4), as we had to slow down to pilot the drone safely closer to the house; at distance 6m. The larger window size of 15s compensates for decelerating during the approach and throughout this period the standard deviation stays above the threshold (recall that our system always chooses the lowest window size that is *above the threshold* and uses it for phase tracking and proximity detection). The surveillance period started at (5) where the drone stayed as still as possible in front of the window. After the rolling windows catch up, the standard deviation drops markedly and stays below the threshold almost the entire time until the escape phase starts at (6). However, it does still breach the threshold occasionally. In practice there are still many

corrections necessary to keep the drone in place in front of the window. As the drone is so close to the receiver during the surveillance phase, even small movements have a notable effect on the RSS and can breach the detection threshold. Finally, the drone landed at (7) leading to another short spike as it reached the ground.

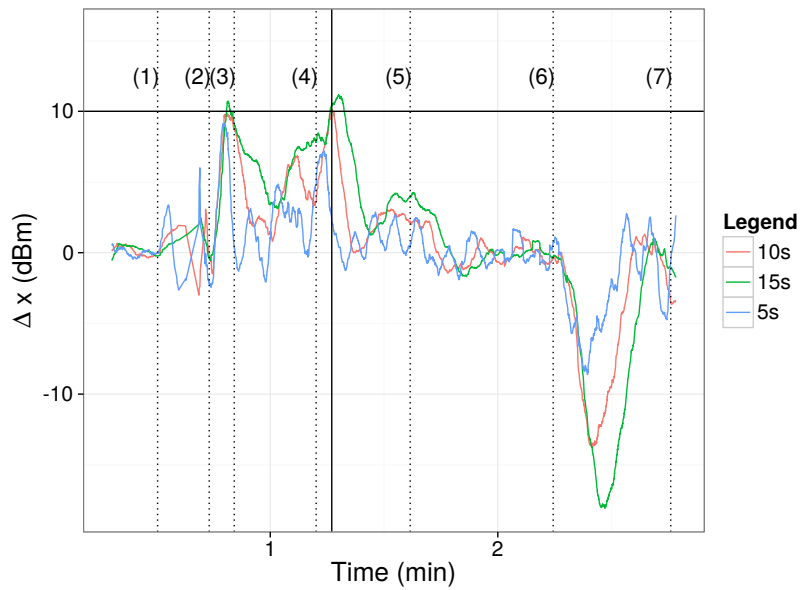
For attack-phase tracking, we compare the first and second halves of w_l , shown in Figure 4.13(b). Both the 10s and 15s windows stay positive for the entire approach. There is more variation in the 5s window as it captures slowing down better than the longer window sizes. All the windows have values close to zero during surveillance after catching up and are negative during the escape phase, as per our expectations.

A proximity alert is triggered when the phase tracking breaches threshold σ_p . In Figure 4.13(b), one can see that with the larger window size of 15s, readings break the threshold early due to the cumulative effect of positive readings over the preceding period. However, at this moment the 15s window is not the smallest for which the standard deviation is above the threshold and therefore another window size is favored over it for phase tracking and the proximity metric—so no spurious proximity warning is produced. The second peak, just after (4), results in a genuine proximity alert for the 10s window. The distance from the window at the time was 4.2m, which is greater than d_s . In Figure 4.13(a), the black vertical line shows the moment when values in the 5s window fall below the detection threshold. Consequently, the 10s window is used to check for proximity and the alarm is triggered.

As mentioned before, we evaluate the free-space propagation test that uses w_s using the Bebop series due to the higher observed packet rate. Figure 4.14 shows the RSS standard deviation for the first straight Bebop run. The standard deviation of the 0.1s window stays mostly below the threshold throughout the approach; only very few samples violate it initially. When the drone is close to the wall of the building, the standard deviation increases even within w_s , hence breaking the threshold more often. At this time, however, we can already be quite sure that it is a drone. The peaks during the surveillance period are caused by movement in front of the window; we had to make more corrections to account for drift with the Bebop, than with the heavier Phantom.



(a) Standard deviation of the signal strength. The black horizontal line is the movement detection threshold.



(b) Attack phases. The drone is approaching when the values are positive and escaping for negative values. The black horizontal line is the proximity threshold.

Figure 4.13: Straight approach with the DJI Phantom 3. The black vertical line shows when the 5s window undercuts the movement threshold. The dotted vertical lines indicate: (1) Takeoff, (5) Surveillance, (6) Escape and (7) Landing. More details can be found in the text.

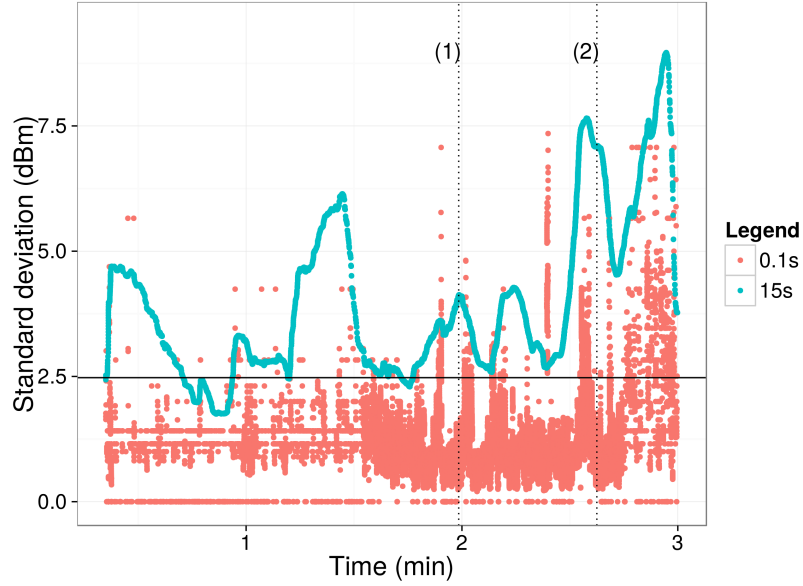
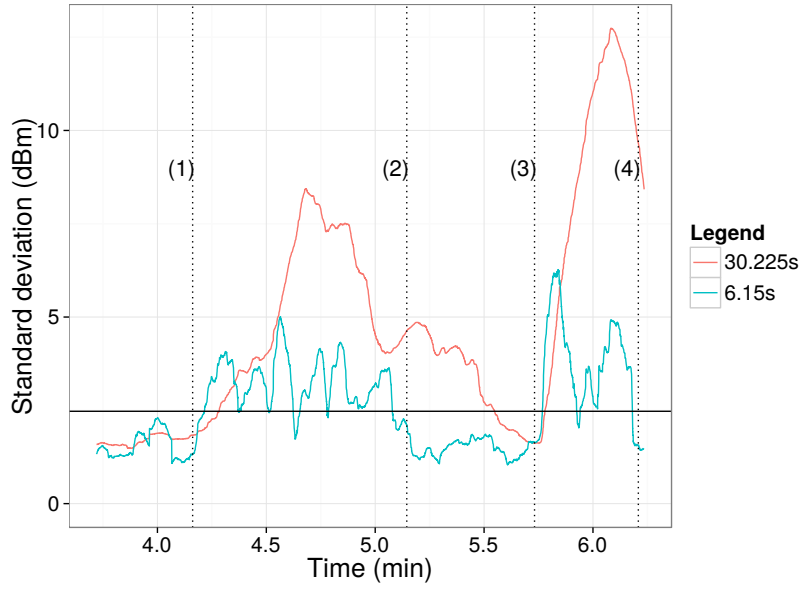
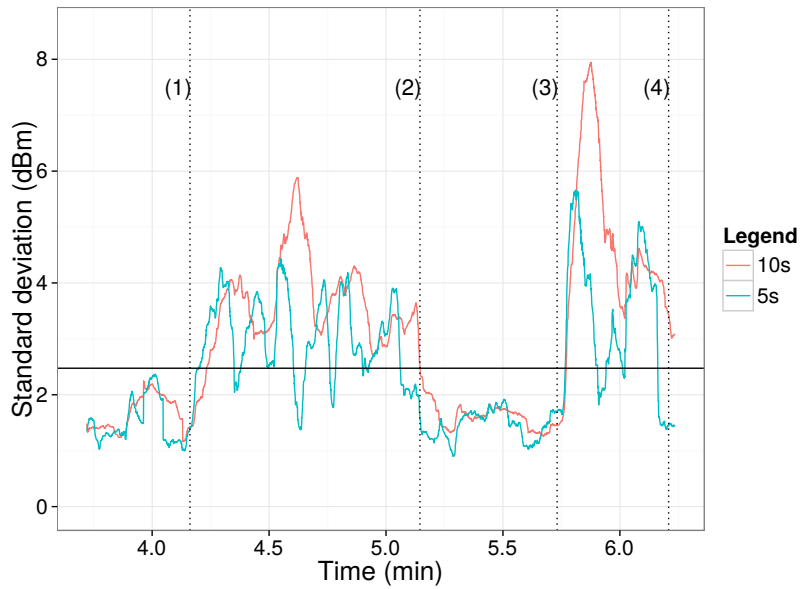


Figure 4.14: Standard deviation for a straight approach with the Parrot Bebop. The black horizontal line is the free-space propagation detection threshold which should not be exceeded within $w_s = 0.1s$. The vertical lines indicate: (1) Surveillance and (2) Escape. More details can be found in the text.

4.7.2 Zig-zag Approach

For the zig-zag approach we will first look at window sizes derived from the maximal and effective approach velocity. For the first zig-zag run the maximal approach velocity was $v_{\max} = 5.12\text{m/s}$. Using $v_{\max}$ we compute $w_{l,\max} = 6.15\text{s}$. On the other hand, the effective velocity for the whole approach was only $v_{\text{eff}} = 1.04\text{m/s}$ and thus $w_{l,\text{eff}} = 30.225\text{s}$. In Figure 4.15(a) one can see that the smaller window size detects the drone faster at a distance of 59.5m whereas the larger window size only detects the drone 53m away from the receiver. However, $w_l = 6.15\text{s}$ captures the erratic changes caused by the zig-zag movement, whereas $w_l = 30.225\text{s}$ smooths out the flight pattern and shows the whole approach. The decrease in standard deviation in the second part of the approach is caused by a general decrease in approach speed.

We can see the same effect if we use our system with the parameters from the beginning of the section, *cf.* Figure 4.15(b). Lower window sizes than the derived values are possible as $w_l = 10\text{s}$ already captures the complete approach. This is again likely to be due to an increased standard deviation because of the launch and acceleration periods.

(a) Parameters derived from the model using $v_{\max}$ and v_{eff} 

(b) System parameters

Figure 4.15: Standard deviation for a zig-zag approach with the DJI Phantom 3. The black horizontal line is the movement detection threshold. The vertical lines indicate: (1) Takeoff, (2) Surveillance, (3) Escape and (4) Landing. More details can be found in the text.

4.7.3 Back-and-forth Approach

For the back-and-forth approach, we proceed in a similar fashion as for the zig-zag approach. We compute w_l for the maximal speed $v_{\max} = 4.24\text{s}$, and the effective velocity $v_{\text{eff}} = 0.72\text{s}$. Similarly to before, Figure 4.16(a) shows that the former is affected by the flight pattern whereas the latter captures the complete approach. The shorter window $w_{l,\max} = 7.425\text{s}$ shows larger peaks followed by smaller peaks. The larger peaks correspond to forward, the lower peaks to backward motion.

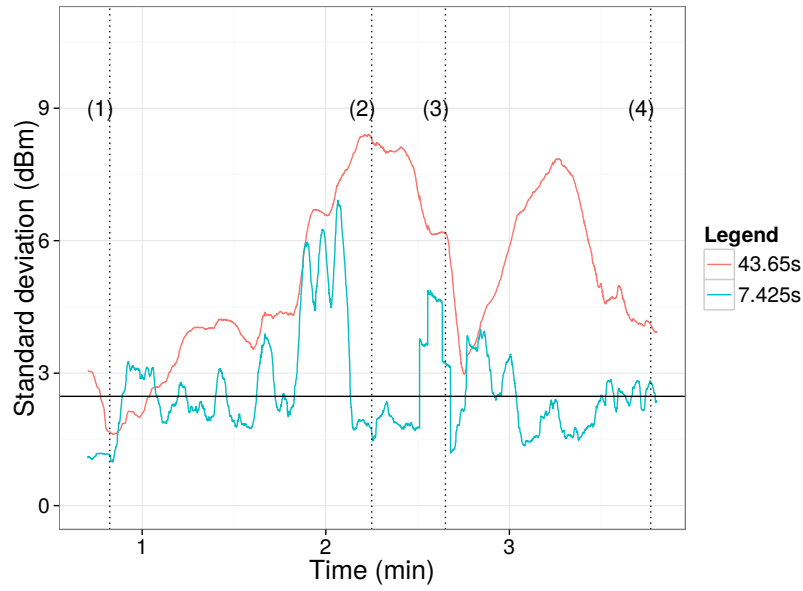
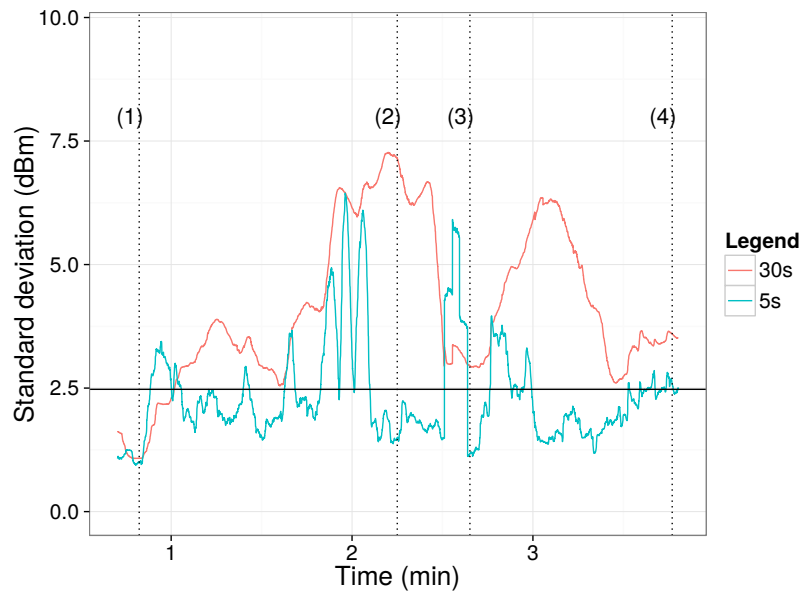
In comparison, the higher window size $w_{l,\text{eff}} = 43.65\text{s}$ is able to capture the complete approach as a whole. However, it also covers the entire surveillance period, again highlighting the need for the proximity metric as a stop-and-alarm condition. If we use the system parameters as before, we can see in Figure 4.16(b) that the 30s window is sufficiently large.

Additionally, we can study the approach phase tracking in Figure 4.17. We can see that if we only rely on the shorter window, the system oscillates between approach and escape phases, whereas the longer window size makes it possible for us to detect the approach as a larger trend.

4.7.4 NLOS Approach Over the Roof

The first NLOS run with an approach over the roof is pictured in Figure 4.18(a). The drone takes off at launch distance 28m on the other side of the house. Its forward acceleration starts at (2) and it reaches maximal altitude at (3). At (4) the descent begins, on the receiver side of the house. The first big peak for window size 15s happens as the drone is flying over the receiver, above the roof. The next peak occurs as it is descending into LOS, with another one following when it gets closer to the window up to d_s . The surveillance period starts at (5), before the drone escapes over the roof again at (6).

We had to fly at slow speeds close to the house for careful maneuvering which explains why higher values are needed to capture the whole approach compared to the straight LOS series.

(a) Parameters derived from the model using $v_{\max}$ and v_{eff} 

(b) System parameters

Figure 4.16: Standard deviation for a back-and-forth approach with the DJI Phantom 3. The black horizontal line is the movement detection threshold. The vertical lines indicate: (1) Takeoff, (2) Surveillance, (3) Escape and (4) Landing. More details can be found in the text.

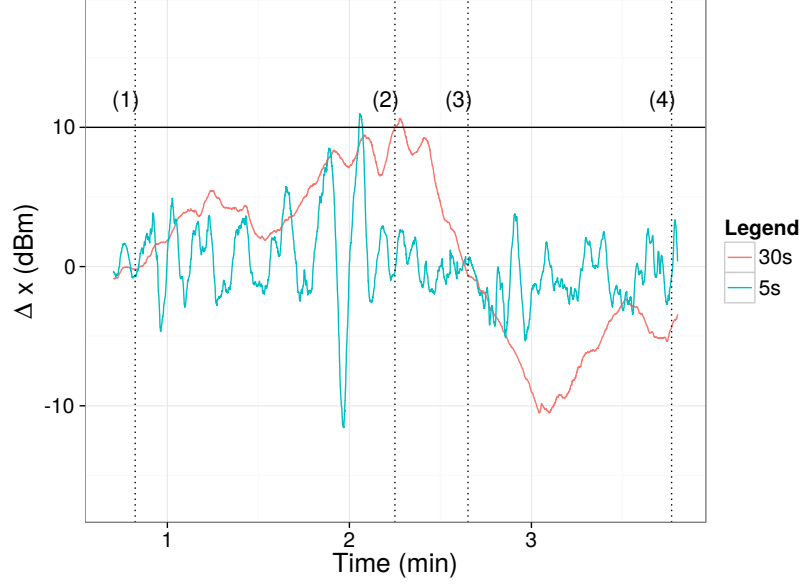


Figure 4.17: Attack phases for a back-and-forth approach with the DJI Phantom 3. The black horizontal line is the proximity detection threshold. The drone is approaching when the values are positive and escaping for negative values. The vertical lines indicate: (1) Takeoff, (2) Surveillance, (3) Escape and (4) Landing. More details can be found in the text.

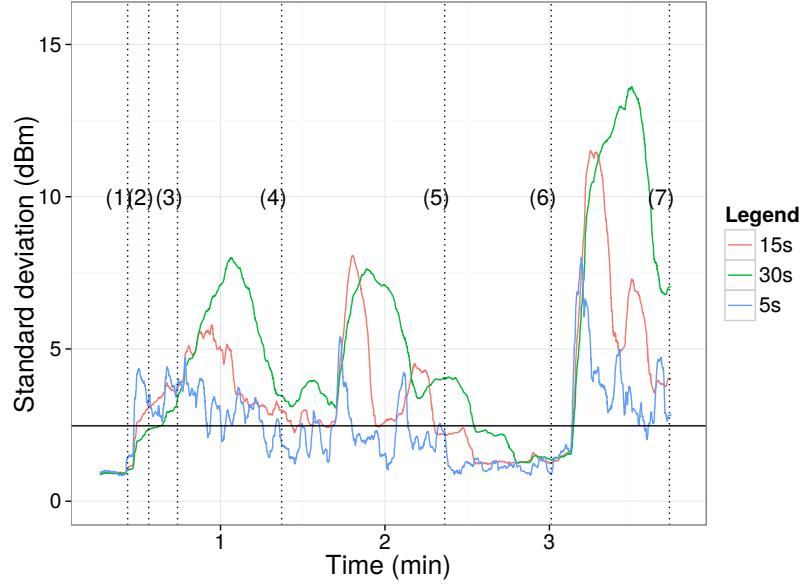
In Figure 4.18(b), the standard deviation within w_s is displayed. One can see that the standard deviation is low in spite of the missing LOS connection. The roof of the house seems to affect the signal only by attenuation and does not lead to significant interference due to multipath effects.

4.7.5 NLOS Approach Around the House

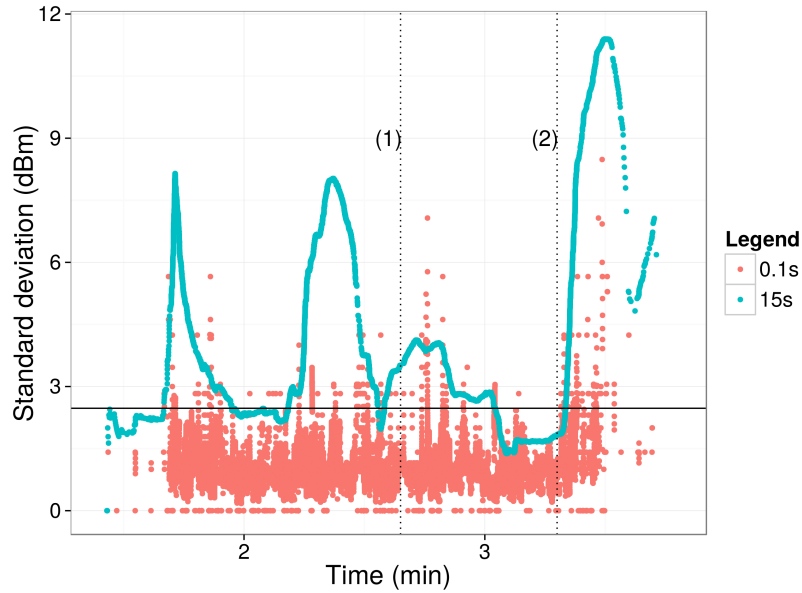
A run of the other NLOS approach variant, around the house, is shown in Figure 4.19. Similarly to the approach over the house, we see peaks as the drone crosses the receiver, gets into LOS and approaches the window. For both NLOS approaches there is a distinct increase in the standard deviation when the drone comes into LOS. Once it is in LOS the behavior is similar to the LOS approaches.

4.7.6 Detection Ranges

To examine detection range we let several rolling windows (5, 10, 15, 30 seconds) run in parallel and chose the first window size to detect the drone as well as the



(a) Movement test with DJI Phantom 3. The vertical lines indicate: (1) Takeoff, (5) Surveillance, (6) Escape and (7) Landing. More details can be found in the text.



(b) Free-space propagation test with Parrot Bebop. The vertical lines indicate: (1) Surveillance and (2) Escape. More details can be found in the text.

Figure 4.18: Standard deviation for a NLOS approach over the roof. The black horizontal line is the noise threshold.

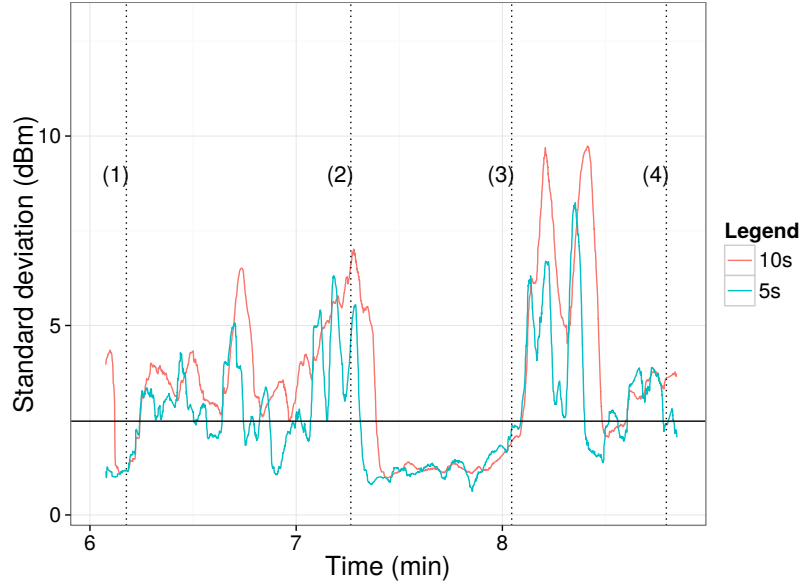


Figure 4.19: Standard deviation for a NLOS approach around the house with the DJI Phantom 3. The black horizontal line is the movement detection threshold. The vertical lines indicate: (1) Takeoff, (2) Surveillance, (3) Escape and (4) Landing. More details can be found in the text.

first that stayed above the threshold for the whole approach phase.

Table 4.4 shows the resulting movement detection ranges. In all of the LOS runs, our system detected the drones far earlier than the minimal detection range of 24.5m, as measured from GPS traces using Google Earth. The detection distances of the second and third back-and-forth runs are lower than the first as they began with a longer and faster forward movement and thus still resembled a straight approach at the time of detection. That the drone flew the approaches with varying velocity actually increased the detection range, as did the more time-consuming flight patterns.

The proximity alert was triggered within 5m of the target window in every experiment run. Unfortunately, due to the roof occluding GPS satellites, the data from the traces is too imprecise to determine the exact distance in each case.

Table 4.4: Detection ranges of the DJI Phantom 3 LOS experiment series.

Approach pattern	Detection range (m)			
	Run	1	2	3
Straight		55.5	55	53.77
Zig-zag		60	59.5	60.13
Back-and-forth		60.22	51.47	48.65

4.8 Discussion of Preliminary Results

4.8.1 Results

The most notable discovery throughout our experiments was that detection happened earlier than our simulations had predicted. In large part this is simply attributable to our model being very conservative and real-world attacks not demonstrating such extreme behavior. Ever-changing conditions and operator inaccuracy leads real drone flight to have far more speed variance than the constant-speed approaches we had assumed. No two consecutive flights were identical, even with the same intended approach pattern. Close to the target, in the latter stages of an approach and during surveillance, the movement is even more erratic as the drone must navigate a tighter environment and counteract drift from wind conditions and sidewash turbulence in order to establish and maintain a close hover. The reliance purely upon the operator's precise control is intensified as GPS stabilization systems are hampered by a reduced view of the sky when near a building. The detection system parameters must be carefully selected to ensure that the Escape phase is not indicated prematurely.

Takeoff is the first movement a drone makes, but communication begins before this point. This prior communication is beneficial to our detection as the received messages establish an initial measurement in each window, such that the takeoff itself is enough to bring the standard deviation above the detection threshold in many cases. Such early detection gives ample time for the user to be warned of a privacy-invasion attack and take protective action. It also presents an opportunity to simply inform a user of nearby drone activity that is not, or has not yet developed into, an attempt at privacy invasion.

4.8.2 Observations

Our experimental experiences reinforced the expectation of a reliance by the pilot on the drone’s FPV video stream. At close range it is possible (and sometimes preferable) to watch the drone directly when flying, however the benefits of doing this quickly disappear as the drone operates further away and the pilot is less able to reason about its position relative to other objects purely by eye. In this case, and certainly where there is no direct line of sight at all, the attacker depends on the streamed video to pilot successfully. Even in the minimal case, conducting the privacy attack itself, the operator needs immediate feedback to ensure that the drone obtains a good view of the interior of the building—it proved easy to capture detailed footage of a window-frame by mistake. An attacker could, if their hardware permitted, disable the video stream for a period in order to avoid providing a suitable communication stream for the detection system. However, doing so near the target or when attempting surveillance, would likely jeopardize the attack. Indeed, a short surveillance distance is crucial in mounting a successful attack; at large distances a high-resolution camera or telescopic lens is required to capture a detailed image of the building interior and even then the visible area is heavily constrained by the window. Our observations suggest that values of d_s at one or two meters are realistic.

If the drone flies close to the ground for the entirety of the approach, ground reflections are strong multipath components and violate the free-space propagation assumption—making the drone indistinguishable from an attacker moving on the ground themselves. In practice however, we expect that the attacker has to overcome access restrictions and hence must fly higher for at least some part of the approach, making the drone detectable again.

4.8.3 Deployment Considerations and Mobile Transmitters

The proposed standard deviation-based drone detection method uses one COTS device to detect incoming commercial drones based on their signal characteristics. This device could very well be a smart home device sharing its RSS observations with the rest of the smart home network. However, this method is constrained to

using a single device at a time and relies on specific placement of the receivers used for detection. This means that the system is leaving a multitude of devices already present in modern smart homes unused and thus misses out on potential performance boosts. Furthermore, there is a lack of certainty regarding its performance when faced with cross-traffic of various static and ground-based moving transmitters. Nowadays, smart homes have busy radio environments due to the multitude of devices communicating wirelessly. These devices range from routers, PCs and other static devices, to devices carried by the user such as laptops or smartphones. The standard deviation-based method can be expected to cope well with static transmitters (*e.g.*, desktop PCs, routers or laptops used in a stationary manner), as variation in their signal strengths is largely due to multipath effects.

However, it remains unclear whether this method might confuse moving transmitters—especially those outside the house—with drones, as moving transmitters are much more likely to exhibit the characteristics assumed by the model. It is therefore important to consider these types of non-drone transmitters when designing an effective drone detection system. Therefore, we perform a new set of experiments to determine whether the system is prone to false alarms in the presence of (benign) static and ground-based mobile transmitters. As part of these new experiments, we also investigate how less ideal device placements affect the detection.

4.9 Experiment with Mobile Transmitters and Interfering Traffic

We conduct the new experiment in the same place and using the same approach patterns as the preliminary experiment described in Section 4.6 in order to ensure comparability. In this new experiment we consider a wider range of receiver placements including less ideal placements without LOS to the approaching drone. Furthermore, we consider both static and mobile benign transmitters in order to investigate likely sources of false alarms.

The setup of the experiment, placement of the devices and the routes of the different non-drone approaches can be seen in Figure 4.20. For the receivers,

we used eight Raspberry Pi 4s with a two-antenna Wi-Fi card (Alfa Network AWUS036AC) attached over USB. The receivers were constantly recording Wi-Fi traffic in monitor mode. The drones, static transmitters and the smartphone used as a moving non-drone transmitter were all configured to communicate on the same Wi-Fi channel in the 2.4 GHz band.

Four of the eight receivers (RX 1–4) were placed in windows facing the direction from which the LOS drone approaches were launched. The other receivers were placed in common places around the house: in a bedroom at the back of the house (RX 8), in the downstairs living room (RX 5), at the top floor ceiling similar to a smart light bulb (RX 7), and in the kitchen next to the Wi-Fi router (RX 6).

4.9.1 Non-drone Approaches

The non-drone approaches were done using a Huawei P20 Pro smartphone running an iperf2 compatible app creating UDP traffic with a packet size of 750 bytes and a bandwidth of 1.5 Megabytes/second. We considered seven different non-drone approach patterns in which the phone was carried in the hand of one of the authors. These patterns are listed in Table 4.5 and their paths are outlined in Figure 4.20. They include entering and exiting the house from the front and the back, starting at a distance of 15m from the house and tracking the drones' approach paths on the way. These patterns were included as they closely resemble a drone approaching the house from the respective side of the house and might therefore be difficult for the system to classify correctly. In addition to these four patterns, we defined three movement patterns inside the house. For each floor we defined a pattern that includes approaches to every receiver and covers every room in the house. Finally, one pattern involved walking down the hallway and going up and down the stairs. We completed at least five runs of each approach pattern.

4.9.2 Static Transmitters

For the static transmitters, we used seven Raspberry Pi 4s that were constantly transmitting traffic to a server attached to the router via ethernet. The traffic was

Table 4.5: Non-drone approach patterns used in experiment. These patterns were completed using a smartphone. The receivers in LOS of the phone varied between approach patterns.

Approach	Description	Receivers in LOS
Front door enter/exit	Enter/exit house through front door	RX 3, 6
Back door enter/exit	Enter/exit house through back door	RX 1, 2, 3, 4, 6
Ground floor	Approach each transmitter on the ground floor	RX 1, 3, 5, 6
First floor	Approach each transmitter on the first floor	RX 2, 4, 7, 8
Stairs	Walk down hallway and up/down the stairs	RX 7

generated using iperf2 with the same parameters that were used with the phone for the non-drone approaches (*cf.*, Section 4.9.1). By running the static transmitters while the non-drone approaches were performed, we ensured that there was data from a dynamic environment with sufficient line breaks for each transmitter/receiver pair. The static transmitters were placed in locations where one might find high-bandwidth traffic sources (*cf.*, Figure 4.20): In the living room, like a voice assistant or smart display, *e.g.*, Google Home or Amazon Echo Show (TX 10), in two of the bedrooms, like a laptop or desktop PC (TX 6, 8), and at the front and back door similar to smart door bells with integrated cameras (TX 1, 2).

4.9.3 Drone Approaches

For the drone approaches, we used the same four approach types as in Section 4.5.3 (*cf.*, Table 4.1), in each case using the first floor window with RX 2 as the privacy-invasion target. As done in the previous experiment, the NLOS approach was further split into approaches over the roof and around the house to include both a descent from above and reaching the window from the side. Every approach in the series was performed with at least five run repetitions for each of the two used drones (DJI Phantom 3 Standard and Parrot Bebop). The launch distance was between 42 and 50 meters for the LOS approaches. For the NLOS approaches the

Table 4.6: Results for the standard deviation-based method from Section 4.5. Numbers in parentheses represent the absolute number of missed drone approaches for DR, or false alarms for FAR.

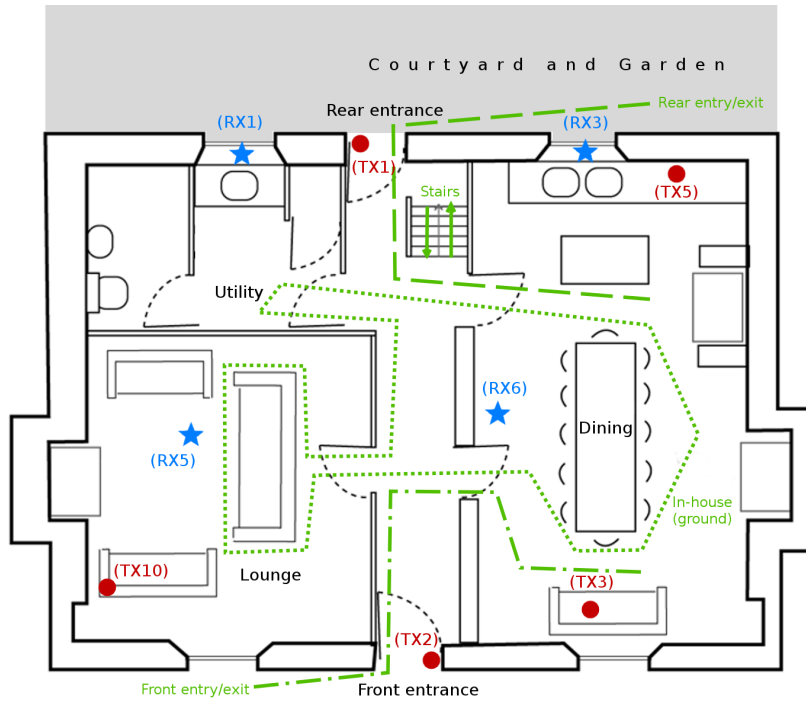
Receiver	DR	FAR (static)	FAR (moving)
1	0.98 (1)	0.02 (6)	0.381 (16)
2	0.961 (2)	0.041 (12)	0.619 (26)
3	0.961 (2)	0.122 (36)	0.571 (24)
4	0.961 (2)	0.031 (9)	0.357 (15)
5	0.784 (11)	0.085 (25)	0.5 (21)
6	0.843 (8)	0.095 (28)	0.595 (25)
7	0.823 (9)	0.068 (20)	0.524 (22)
8	0.803 (10)	0.024 (7)	0.643 (27)

launch distance was much shorter at around 30 meters as they started from the front side of the house, with the operator moving to keep the drone in view during the approach. No run exceeded a top speed of 7m/s (approx. 25kph/15mph), as it is hard to control the drone at high speeds close to a building.

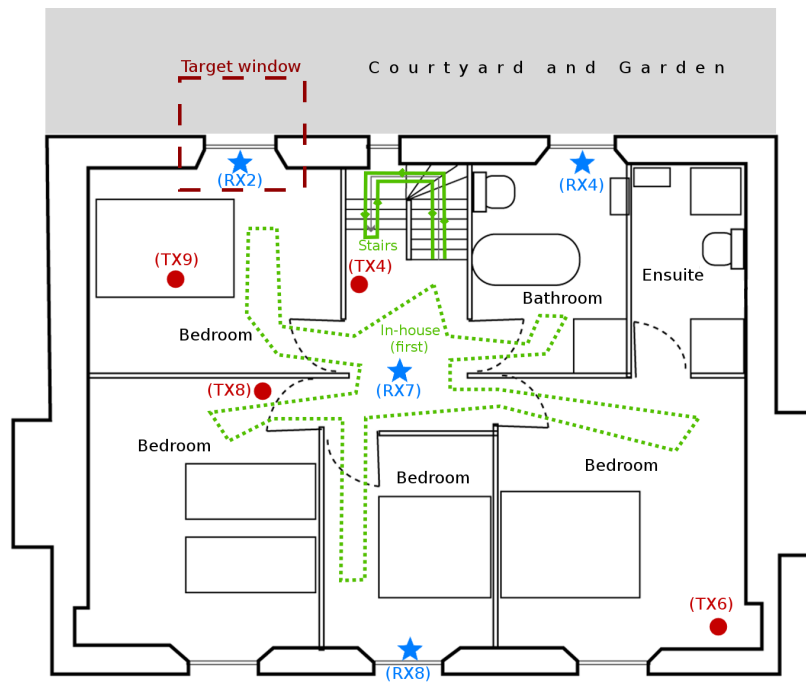
4.10 Evaluation of the Standard Deviation-based Method

Using the data gathered in this extended experiment, we evaluated the method described in Section 4.5 by considering static and moving transmitters separately, as well as by taking different receiver locations into account. The evaluation is done for each receiver independently. In this evaluation, each sample of the positive set is a run of a drone approach pattern using the data originating from the approaching drone. By including every approach pattern for both drones this sums up to 51 samples in the positive set. The standard deviation-based method achieves a high detection rate (DR) for all of the receivers situated in one of the windows facing the approaching drones. The maximal DR is 0.98 at receiver RX 1 and none of the window receiver misses more than 2 drone approaches. The results for all the receivers can be found in Table 4.6.

For the static transmitters, each sample in the negative set is a run of a non-drone approach pattern using the received data of one of the seven static transmitters.



(a) Ground floor



(b) First floor

Figure 4.20: Approach patterns and equipment placement. The target window of the intruding drone is indicated by a red dashed box. The receiver locations are marked with blue stars, whereas the static transmitter locations are denoted with red dots. Green lines indicate approach patterns.

In total, this sums up to 294 samples in the negative set. We only consider the time of non-drone approaches for the negative set as this was the only time during the experiment with significant activity within the house. Because the static transmitters are only expected to lead to false alarms in a dynamic environment as people move between receivers and transmitters and thus change the signal environment, considering the whole duration of the experiment would skew the results towards a lower false alarm rate (FAR). Nevertheless, only few false alarms are registered in the case of static transmitters with an FAR of 0.041 or 12 false alarms measured at receiver RX 2 in the target window. This indicates that this method is well suited to distinguish drones from static transmitters.

However, this is not the case for other moving transmitters. The negative set for the moving transmitters consists of all the runs of non-drone approach patterns using the data originating from the smartphone. The total size of this set of samples is 42. The performance is far worse in this comparison against moving non-drone transmitters. We measured an FAR of 0.619 at receiver RX 2 in the target window with a total of 26 false alarms being registered.

4.11 Kurtosis-based Method

Building on the insights that we gained from the results of both experiments and to address the issues of the previous method, we propose an advanced drone detection method that uses improved statistical metrics to better capture the difference between airborne and ground-based moving transmitters. In contrast to the previous method, this new method does not require control over the deployment of the used devices and is additionally able to leverage several devices at once. Hence, the whole smart home network can be exploited to improve the detection performance. This means that the new method does not require the devices to be repurposed for their new security function and thus signifies an important step towards the use of ubiquitous sensing for security.

This improved method is based on the kurtosis of short-term signal changes and stems from the observation that we expect fewer outliers for the signal strength

measurements of the drone compared to other transmitters due to the more consistent radio environment and the stable LOS to the receiver in the case of the ideal placement. The kurtosis is the fourth standardized moment and measures how far samples are from the mean. Samples within one standard deviation of the mean contribute almost nothing to the value of the kurtosis, whereas samples that are further away have a higher impact based on distance from the mean. Therefore, a higher kurtosis means that there are more extreme outliers with respect to the standard deviation.

4.11.1 Long-term Test

We combine the proximity and approach phase test into a single test. This test triggers when the drone moves from the launch distance to a pre-defined detection distance. For this test we check if the difference of the median of the last few samples within a rolling time window of length w_l and the median of the first few samples breaches a signal threshold Δ_p that can be derived from the launch distance d_l and the detection distance d_d . We use the median to be more robust to singular outliers. The proximity and approach phase test can thus be defined as follows:

$$\text{median}(\text{last } 100ms) - \text{median}(\text{first } 100ms) \geq \Delta_p$$

where

$$\Delta_p = 20 \cdot \log_{10}(d_l) - 20 \cdot \log_{10}(d_d)$$

In addition to this, we use a similar movement test to before, using the sample standard deviation of the N samples captured within the rolling window. Because of the integrated proximity test, we adapt the method from Section 4.5 to account for the expected movement of the drone from the launch distance d_l to the chosen detection distance d_d .

We know from Section 4.5 that $\text{Var}(FSPL + X_N) = \text{Var}(FSPL) + \text{Var}(X_N)$ and $\text{Var}(X_N) = \sigma^2$. Here, σ is the noise threshold derived according to Section 4.5.

$\text{Var}(FSPL)$ is given by:

$$Var(FSPL) = Var(\{20 \cdot \log_{10}(d_l - i \cdot (d_l - d_d)/N) \mid \forall i \text{ in } 1..N\})$$

Then the movement test succeeds if:

$$s(N) > \sqrt{Var(FSPL) \cdot \sigma^2}$$

For this combined long-term test, several rolling window lengths are used to account for different drone speeds. At each point in time, the window that minimizes the value used in the short-term test described in the following subsection is used for the drone detection.

4.11.2 Short-term Test

In the short-term test, we are interested in signal strength changes that are not related to the distance between the drone and the receiver, but are instead due to small-scale fading effects. To minimize distance related changes, we consider the difference between subsequent samples. This is motivated by the observation that the difference in free-space path loss of two consecutive samples is negligible if the sampling rate is sufficiently high.

The sample kurtosis of the N samples captured within the rolling window is as follows:

$$k(N) = \frac{(\frac{1}{N} \sum_{i=1}^N \dot{x}_i - \bar{x})^4}{s(N)^4}$$

Here, a sample $\dot{x}$ is defined as the difference of two consecutive samples:

$$\dot{x}_i = x_i - x_{i-1}.$$

We consider the signal strength of the drone signal as the sum of random variables $FSPL$ for the free-space path loss and X_N for the noise (*cf.*, Section 4.5). Then the sum of these random variables for the difference of consecutive samples is given by:

$$\dot{X} = X_i - X_{i-1} = FSPL_i + X_{N,i} - FSPL_{i-1} - X_{N,i-1} = X_{N,i} - X_{N,i-1}$$

As the free-space path loss cancels out, the difference of consecutive samples is the sum of the noise of both samples. If we assume that the noise of each

sample is normally distributed, it follows that the difference also follows a normal distribution, as the sum of two normal random variables remains normal [198]. Therefore, we expect the kurtosis of the sample differences to have a kurtosis close to three (excess kurtosis close to zero). We expect higher values for the kurtosis of other moving transmitters as they will be subject to other sources of noise and are thus more prone to having outliers.

4.11.3 Message Rate

As modern drones are streaming high-quality live video to enable responsive control, a high packet rate can be expected. Therefore, we test if the packet rate is high enough within the tested time period. However, there can be a large amount of packet loss if the drone is still further away. Therefore, this threshold has to be chosen carefully to trade-off detection performance with detection speed.

4.11.4 Algorithm

In order to detect drones, we compute the combined approach/proximity and movement tests described in the previous subsections. This is done using sliding windows with increasing values of w_l . When the signal strength increases by more than Δ_p and if there is noticeable movement as signified by the standard deviation of the signal, we check if the message rate is high enough. Once all these tests are positive, we finally check if the kurtosis of short-term signal differences is lower than the threshold Δ_k . If this is the case, a drone alarm is raised, otherwise the system resumes operation. Pseudo-code for the detection algorithm can be found in Algorithm 2.

4.12 Evaluation of the Kurtosis-based Method

To evaluate whether the assumption holds that the difference of consecutive drone signal strength samples resembles a normal distribution, we choose an excess kurtosis threshold of 0 for the short-term test. To study the effect of the short-term test

Algorithm 2: Kurtosis-based detection algorithm. Here, s , k and $rate$ are the tests introduced in Section 4.11. A full description for the algorithm is given in Section 4.11.4.

Data: Data flow of one transmitter
Result: Boolean: Drone alarm
begin
 Drone alarm $\leftarrow \emptyset$
 for Increasing w_l until $v_{\min}$ **do**
 if median(*last 100ms*) – median(*first 100ms*) $\geq \Delta_p$
 and $s(w_l) > \sqrt{\text{Var}(FSPL) \cdot \sigma^2}$ **then**
 if $rate(w_l) \geq \Delta_r$ and $k(w_l) \leq \Delta_k$ **then**
 Drone alarm $\leftarrow \mathbf{True}$
 else
 Drone alarm $\leftarrow \mathbf{False}$

and the message rate limitation separately, we first concentrate on the former and disable the message rate check.

We find that we achieve a DR of 0.902 at a detection distance of 30m with an FAR of 0.095 for the receiver in the target window (RX 2). Both FAR and DR decrease with decreasing detection distance; the FAR reaches a value of 0 at $d_d = 20$ with a DR of 0.765, and the lowest DR is at $d_d = 10$ with a value of 0.569.

If only the long-term test is being used then the DR is consistently at 1.0 but the FAR is very high ranging from 0.73 to 0.97.

If we look at the histogram of the kurtosis values for a detection distance of 15m in Figure 4.21, we can see that although the DR is low if a threshold of 0 is chosen, the kurtosis values for both of the drones are still consistently lower than those of the moving phone.

4.12.1 Detection Performance

If we take this observation into account, we can use higher thresholds on the kurtosis to improve the performance of the detection system. Figure 4.22 shows a receiver operating curve (ROC) that compares the performance at the receiver in the target window (RX2) for the cases of static and moving transmitters with different

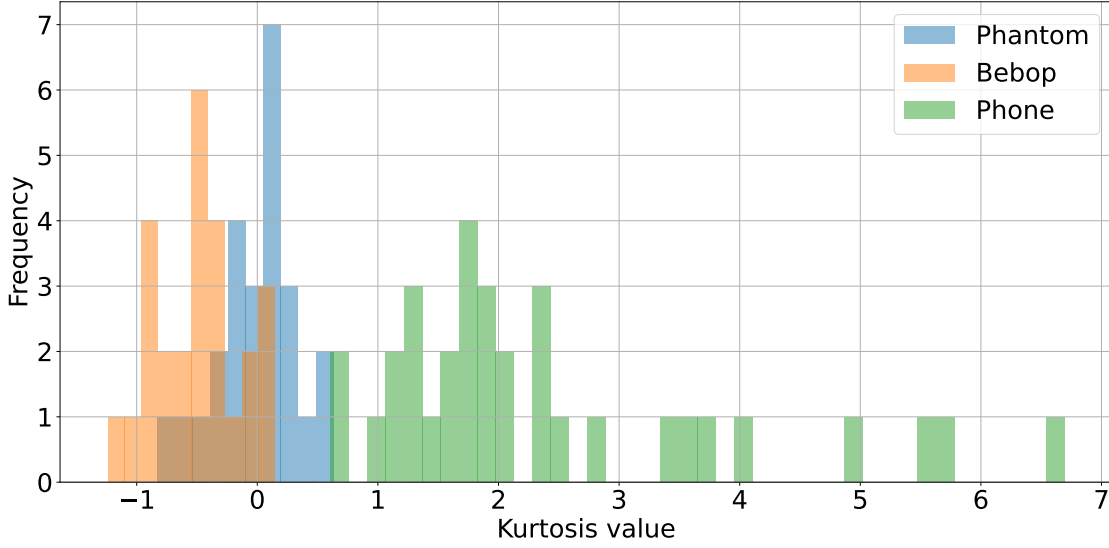


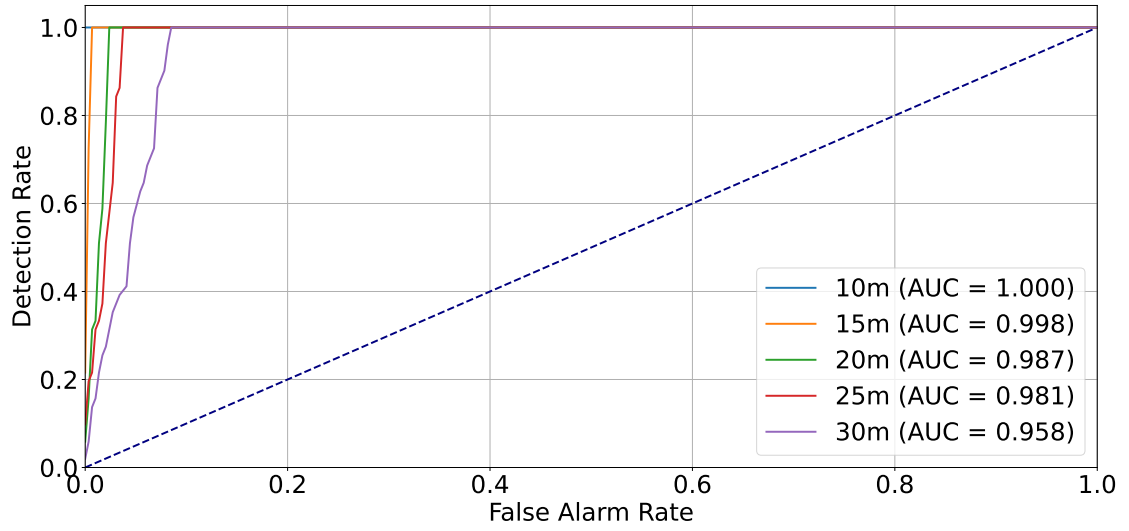
Figure 4.21: Histogram of the excess kurtosis short-term test values for the two drones and the smartphone with a detection distance of 15m at the receiver in the target window (RX 2). One can see that the kurtosis is consistently lower for both drones compared to the phone.

Table 4.7: Detection performance for the receiver in the target window (RX 2) measured by the AUC for different approach subsets.

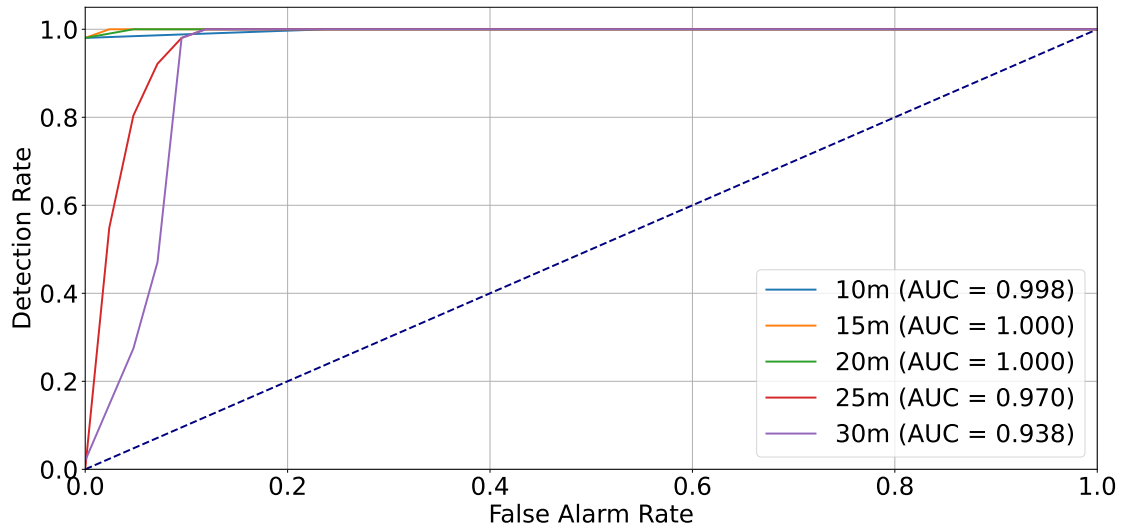
Detection dist. (m)	All app.		LOS app.		Bebop app.		Phantom app.	
	Mov. /	Stat. TX	Mov. /	Stat. TX	Mov. /	Stat. TX	Mov. /	Stat. TX
10	0.998	1	1	1	1	1	0.995	1
15	1	0.998	1	0.998	1	0.999	1	0.997
20	1	0.987	1	0.988	1	0.991	0.999	0.983
25	0.97	0.981	0.974	0.983	0.981	0.987	0.960	0.975
30	0.938	0.958	0.941	0.96	0.951	0.972	0.923	0.944

threshold values. We observe that very good performance can be achieved for both transmitter types, with closer detection distances exhibiting better performance.

Table 4.7 shows the area under the curve (AUC) for the receiver in the target window (RX 2) for different approach subsets. Similar to Figure 4.22, the detection performance is generally higher for detection distances closer to the target window. One can also see that the detection performance increases when only LOS approaches are considered. Furthermore, the system is noticeably better in detecting the Parrot Bebop drone, possibly because of its higher packet rate. Nevertheless, even the performance for the DJI Phantom drone is generally good and no AUC value is below 0.923.



(a) Static transmitters



(b) Moving transmitters

Figure 4.22: ROC curves for various detection distances at the receiver in the target window (RX 2).

4.12.2 Spatial Dependence

The floorplans in Figure 4.23 show the performance for different receiver locations measured by the AUC when moving non-drone transmitters are considered for a detection distance of 15m. One can see that receivers in the windows facing the courtyard generally perform better as the drone launches from there for the majority of the approaches. Hence, these receivers stay in LOS of the drone for the entire approach for the LOS approaches and come into LOS for the NLOS approaches

when the drone reaches the target window. Additionally, receivers on the first floor generally perform better than their ground floor counterparts, presumably because they are subject to weaker ground reflections of the drone Wi-Fi signal.

The receiver in the window on the first floor on the front entrance side (RX 8) exhibits better performance than receivers placed in the middle of the house as it is initially in LOS of the drone for the NLOS approaches.

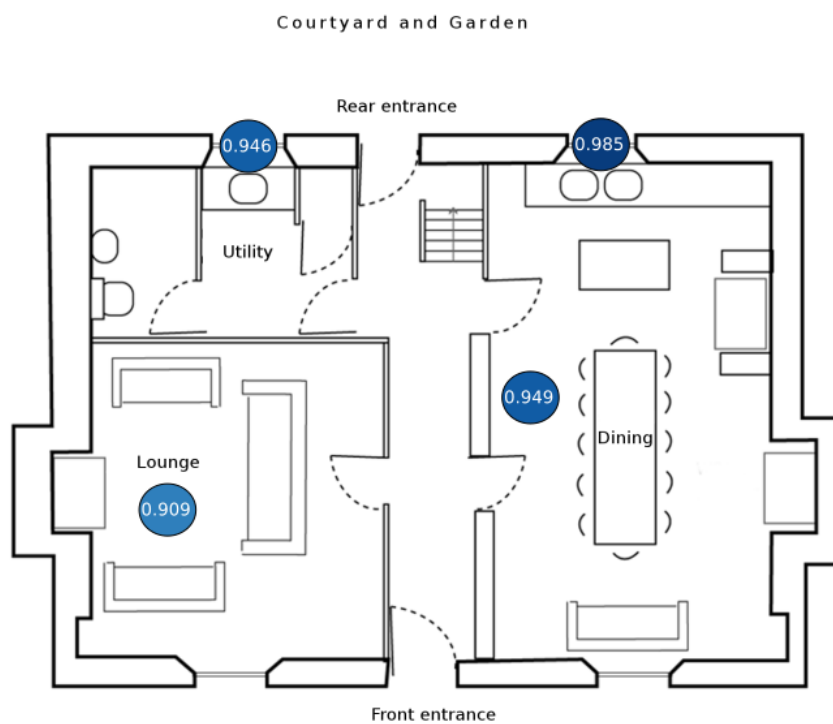
4.12.3 Multiple Receivers

False alarms due to static transmitters are mainly caused by humans moving through the direct line between transmitters and receivers. A line break leading to a false alarm at one receiver is unlikely to cause a false alarm in a different location where the direct line between receiver and transmitter remains uninterrupted. Furthermore, transmitters moving within the house are less likely to keep a stable radio environment between spatially separated receivers. Therefore, we expect that using multiple receivers can improve the detection rate significantly. However, not all of the tests described previously can be applied to more than one receiver at the same time. Tests that include the distance between transmitter and receiver as an implicit factor may fail at one receiver but succeed at the other depending on their spatial separation. Therefore, we use one receiver as the main receiver for the detection, whereas any additional receivers are only used for the long-term movement test.

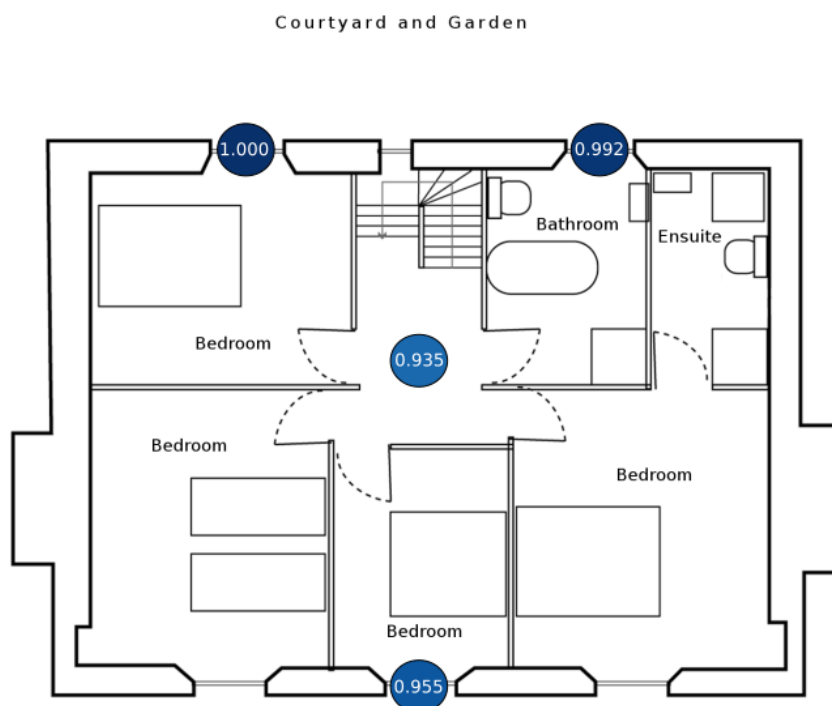
In Figure 4.24, we used the receiver in the target window (RX 2) for all tests and the receiver in the adjacent window (RX 4) only for the long-term movement test. For RX 4 we used the movement test from the standard deviation-based method in Section 4.5 instead of the movement test from the kurtosis-based method as the former test is independent of the distance between receiver and transmitter. There is a considerable performance increase compared to using a single receiver.

4.12.4 Message Rate

The detection performance depends upon the received message rate, which in turn depends on the distance from the receiver. Therefore, a message rate threshold has

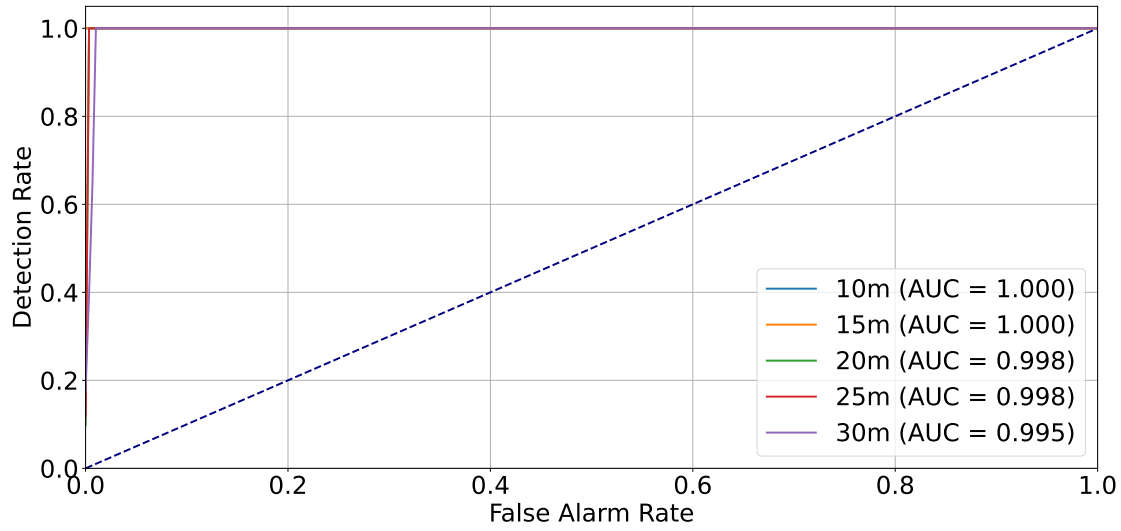


(a) Ground floor

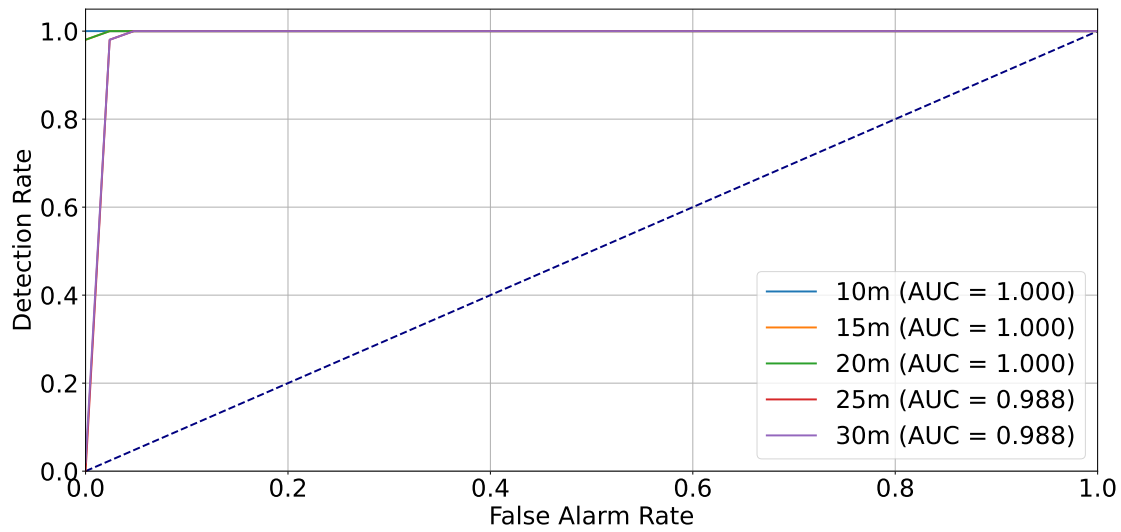


(b) First floor

Figure 4.23: AUC at different receiver locations for a detection distance of 15m.



(a) Static transmitters



(b) Moving transmitters

Figure 4.24: Performance when more than one receiver is used for detection. Here, the receiver in the target window (RX 2) is used for all tests, whereas the receiver in the adjacent window (RX 4) is only used for the movement test.

to be carefully chosen with regards to the desired detection distance. If the threshold is too high, no tests will be performed until the potential drone is very close to the transmitter (*i.e.*, late detection). If the threshold is too low, the computation of statistical measures will be poor (*e.g.*, when computing the kurtosis on very few values). We have selected a nominal minimum message rate during our evaluation and we explore the sensitivity of that selection here.

Figure 4.25 shows the average message rates of the DJI Phantom drone⁸ as received by two receivers (receiver RX 2 in the target window on the first floor and receiver RX 5 in the middle of the living room on the ground floor) at different distances. One can see that the average received message rate for RX 2 is below 20 packets per second when the drone is further than 30 meters away from the target window and only starts to increase noticeably as the drone gets closer than 12 meters to the target. For less than ideally situated receivers such as RX 5, the received message rates are much lower and hardly any packets are received when the drone is further than 30 meters away from the target window.

These observations highlight the need for a carefully chosen message rate threshold depending on the receiver location and the desired detection distance.

4.12.5 Detection Distance

The probability of detection is dependent on how far away from the receiver we want to be able to detect the drone. In Figure 4.26, the detection rate is shown for different distances when the system is tuned for a false alarm rate of zero. This means that for each meter away from the target window, it shows the ratio of drone approaches successfully identified at or before this point when using an optimal set of system parameters for this detection distance. One can see that the system is able to detect all drone approaches eventually without raising any false alarms. LOS approaches are generally detected earlier than NLOS approaches.

Using the receiver in the target window (RX 2), the system is able to detect all drone approaches at a maximum distance of 2 meters (6 meters if only LOS

⁸No GPS data was recorded for the Parrot Bebop in this experiment.

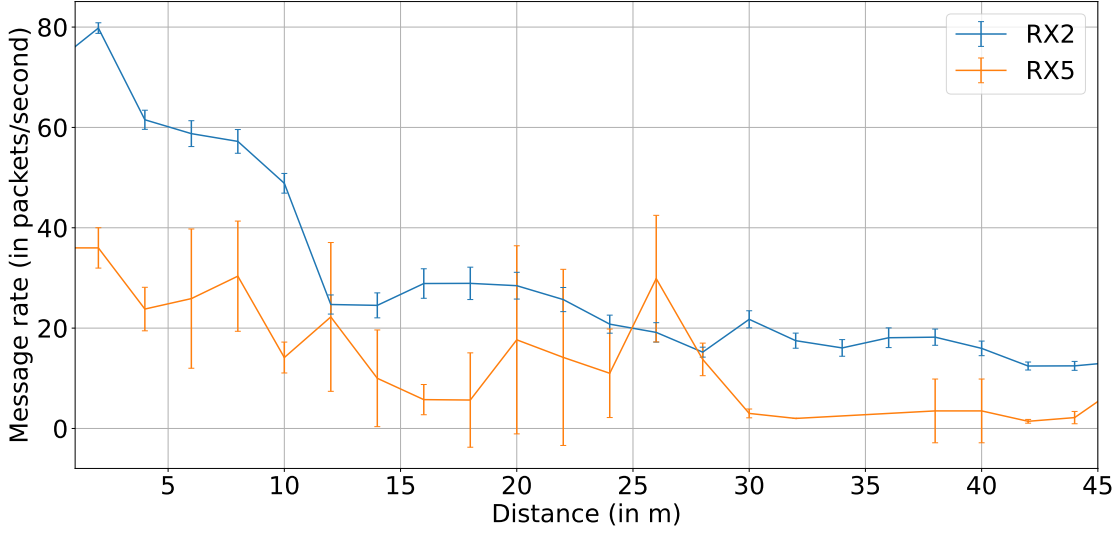


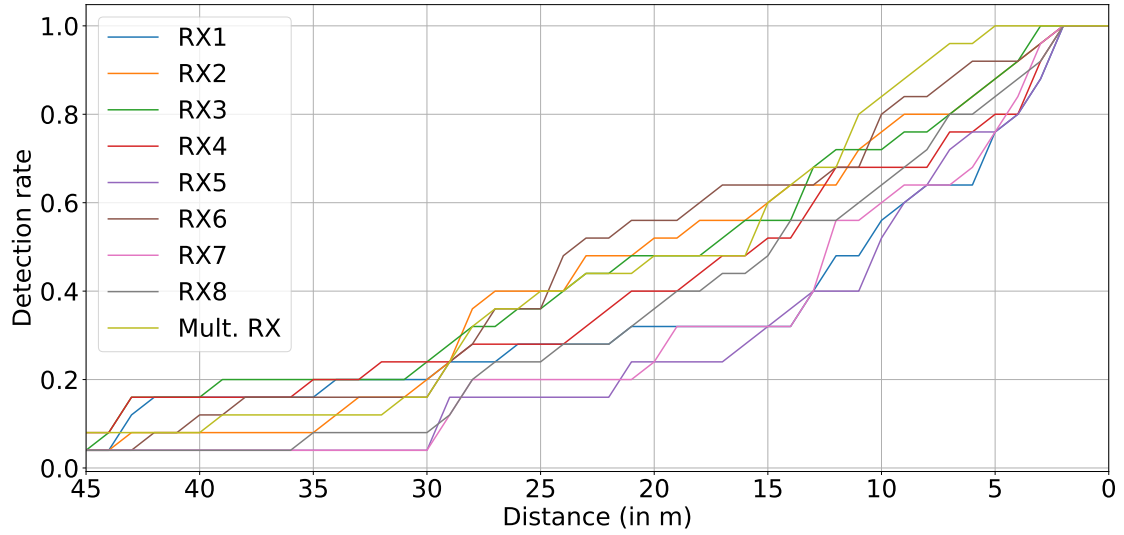
Figure 4.25: Message rates at different distances for the receiver in the target window (RX 2) and the worst-performing receiver located in the ground floor living room (RX 5). Error bars represent the 95% confidence intervals.

approaches are considered). This can be further improved when more than one receiver is considered, with a maximum detection distance of 5 meters for all drone approaches (8 meters for LOS) when RX 2 is used in combination with the adjacent first floor window receiver RX 4. For LOS approaches, the majority of privacy invasion attempts are detected from further than 25 meters away from the target window in both cases (single and multi-receiver).

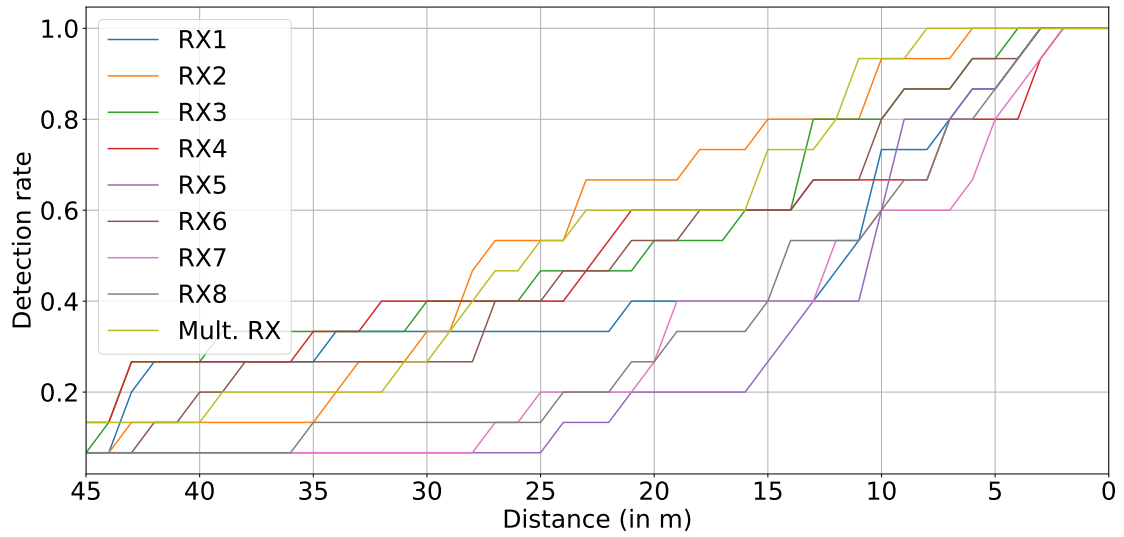
4.12.6 Comparison With Other Systems

A comparison of our approach to other detection systems is challenging due to the many differences in the considered scenarios, experimental design and reported metrics. Out of all the related works, ADS-ZJU [144], Matthan [151] and the work of Nassi *et al.* [152] are the most comparable as they all consider RF-based detection methods, albeit using specialist hardware.

At ranges below 10m, our method outperforms these systems by detecting 100% of drone approaches, with a false alarm probability of 0%. As the system was designed to handle privacy-invasion scenarios, it is unsurprising that it is most effective at these ranges—yet notable as no prior work achieves this performance. The performance reduces with increasing distance however and other systems,



(a) Including NLOS approaches



(b) LOS approaches only

Figure 4.26: Detection rates at different distances with a false alarm rate of zero. Receiver RX 2 is located in the target window and located on the first floor facing the LOS approach site, as does RX 4. Receivers RX 1 and RX 3 are situated below them in the ground floor windows. The other receivers are placed in less ideal positions. The series *Mult. RX* uses receivers RX 2 and RX 4 in the manner described in Section 4.12.3.

typically designed for longer-range detection, then perform better. When our system is tuned to match the metrics quoted in other works it displays: recall of 93.3% versus Matthan’s 97% recall at 10m with a matched precision of 95.9%, detection probability of 73% versus ADS-ZJU’s 100% at 20m with a matched false alarm rate of 2.8%, detection probability of 73% versus Nassi *et al.*’s 100% at 30m with a matched false alarm rate below 10%.

However, our system is purely passive (without an injected visual stimulus as in [152]) and, by contrast, operates with extremely cheap and readily available equipment, such as Raspberry Pis, instead of specialist software-defined radios as all the comparable work does. This means that our system is easier to deploy widely as it can, in many cases, be made available as a software update to devices that are already present in modern smart homes without requiring additional hardware.

4.13 Limitations

In our system, we consider an opportunistic attacker that uses pre-existing COTS hardware to launch an impromptu privacy invasion attack. A more determined adversary with expert knowledge could attempt to evade the system in various ways.

The attacker could try to disable the live video feed and attempt to control the drone by sight alone. However, we have found that it is very challenging to make the necessary precise maneuvers to stay close to the building if the drone is being controlled from a distance by sight alone. This would likely lead to a high risk of crashing the drone. Furthermore, even just having access to the control channel might make detection possible, as the drone is constantly sending telemetry reports back to the controller.

With a pre-programmed flight-path, the attacker could go one step further and disable communication during the attack altogether. But due to the tight maneuvering that is required for the attack and the imprecision of GPS, especially as the drone can lose the connection to some satellites in the shadow of a building, such an attack is unlikely to succeed in close quarters with the target.

Another option for the adversary could be to modify their drone to use a directed antenna in an attempt to avoid exposing the receivers of the detection system to its radio communication. Aside from the additional technical understanding required to perform this modification, beyond simply the purchase of an off-the-shelf drone described in our threat model, this approach has practical difficulties. Using a directional radio link would make the drone harder to control as the controller would have to be consistently lined up behind the drone irrespective of its rotation, or it would require a complicated and expensive tracking mechanism to compensate. Furthermore, as the drone approaches so close to the system receivers, even the weak sidelobes of a directed transmission may still be enough to pick up the drone’s signal effectively.

In an attempt to evade detection, a determined attacker might also modify the drone in a way that allows them to ensure that the signal strength of the drone’s communication is received with a higher variation by the detection system. The attacker could achieve this either by varying the power output of the drone between every packet sent out by the drone, or by introducing an external jamming signal in order to cause signal interference. However, in the case of the former this would require low-level control over the drone’s networking hardware, and in both cases this might reduce the reliability of the drone’s communication with the drone operator.

As our flow separation depends on the availability of consistent MAC addresses and data flows, the attacker could attempt to avoid detection by changing their MAC addresses throughout the approach. We expect this would be effective if the attacker cycles through different MAC addresses with each packet that is sent between drone and controller. However, this requires the attacker to have considerable control over the network stack of the drone, and is not possible with standard consumer equipment. Furthermore, a sudden increase in correlated transmitters could itself be seen as suspicious behavior (but this would warrant further investigation and is beyond the scope of this work).

Finally, we have not tested our system with drones operating in the 5.8 GHz Wi-Fi band. Nevertheless, we expect the detection system to work similarly well

for drones using that band. In fact, due to the increased susceptibility of the 5.8 GHz band to NLOS attenuation and the importance within the system of detecting transition between LOS and NLOS environments, it is possible that the detection may prove more effective in this band.

4.14 Conclusion

In this chapter, we developed a method to detect privacy-invasion attacks by drones based on their communication with a controller. Our approach uses analysis of RSS variance in the drone's transmissions to check for free-space propagation and examine its movement. We developed a further method to monitor if the drone is approaching the detector or moving away from it and another to detect proximity to a receiver. Combined use of these metrics enables us to track the phase of an attack. The system takes into account the challenges posed by cross-traffic of various static and ground-based moving transmitters, as it relies on statistical metrics that capture the difference between airborne and ground-based moving transmitters, thus keeping false alarms low and improving the usability of the system. The sensors used by the system do not have to be deployed with drone detection in mind and the system is able to leverage the observations of several devices to improve detection performance. We conducted real world experiments using two types of commercially-available drones that are controlled over Wi-Fi, with consideration of various example flight patterns and non-drone transmitters. For all series, the approaching drone was detected without causing false alarms for stationary or mobile non-drone transmitters. We were able to detect a drone approaching in LOS of the detection system at a minimal distance of 6 meters, with the majority of approaches detected at 25 meters. Even for a NLOS approach, detection worked reliably and did not inconvenience the user by triggering false alarms.

In summary, our system is able to detect drones flying nearby and can alert when a drone is in proximity of a window. For even the largest of physical windows, the detection happens at a great enough range that an attacker will have had no chance to conduct detailed surveillance before the alarm is raised. Our implementation

used only cheap and easily-obtained hardware representative of devices found in modern smart homes. Moreover, the system is built upon measurements that are available in the vast majority of Wi-Fi capable systems. This means that it can be delivered as a pure software update to existing devices already present in the home—making it easier to deploy in the real world.

5

Smart Home Event Verification in the Presence of Strong Attackers

Contents

5.1	Introduction	99
5.2	Background	103
5.2.1	Smart Home Systems	103
5.3	System Model	104
5.4	Threat Model	107
5.5	Experimental Design	109
5.5.1	Overview	109
5.5.2	Data Collection	112
5.5.3	Ethical Concerns	114
5.6	Methods	114
5.6.1	Data Labeling	114
5.6.2	Dataset Division	116
5.6.3	Sensor Window Selection	117
5.6.4	Feature Extraction	117
5.6.5	Classification and Metrics	119
5.6.6	Attacker Implementation	121
5.7	Results	123
5.7.1	Feature Distinctiveness	123
5.7.2	Spoofing Attack Detection	126
5.7.3	Masking Attack Detection	128
5.7.4	Opportunistic Attacker	129
5.7.5	Sensor-compromise Attacker	132
5.8	Discussion	137
5.9	Conclusion	139

5.1 Introduction

The next important step on the path to creating a system that can successfully leverage ubiquitous sensing for security is to increase the self-reliance of such a system. A self-reliant system needs to be able to autonomously adapt to the specific deployment environment in which it is being used without putting additional strain on the user. In this chapter, we propose an event verification system for event-driven smart home systems that is able to verify the occurrence of smart home events under consideration of both spoofing and masking attacks. We show that this proposed system can learn the necessary selection of sensors and their physical event signatures independently and without supervision by the user, thereby addressing **RQ3**. We choose event verification for event-driven smart home systems as our second example because these systems are the defacto standard for modern smart homes and there is thus a real need to ensure event integrity. Furthermore, event-driven smart homes provide access to event information, which enables the proposed system to adapt to its surroundings without intervention from the user. Due to the insecure nature of many IoT devices, it is however possible that the attacker may be able to read the broadcasted sensor data that underpins the event verification system, or that individual devices used for verification might be under the control of the attacker. When designing security systems based on ubiquitous sensing, it is therefore essential to consider attackers with those kind of capabilities. Therefore, we engage with **RQ4** by demonstrating the impact such strong attackers have on our system.

In recent years, devices that are part of our everyday lives have become increasingly interconnected—creating an Internet of Things (IoT). This has led to the presence of networks consisting of a heterogeneous array of devices in many peoples' homes. Examples of these devices range from surveillance cameras to smart locks, and from remote-controlled garage doors to self-learning thermostats. Due to these devices, users are now able to remotely view their homes, automatically unlock the front door when arriving at home, open the garage door without leaving the car, and offload the burden of worrying about energy-efficient heating. The control of these devices is typically facilitated by event-driven smart home systems, which

Table 5.1: Smart home devices and possible attacker goals. Attacker goals follow the classification by Denning *et al.* [49]. Each device affects the readings of nearby sensors that could be used for event verification. Devices used in our real-world experiment are highlighted in bold.

Device	Attacker goal	Sensors for verification	Ref.
Door	Enabling physical entry, Physical theft	Accelerometer, Light, Air pressure, Microphone	[22]
Window	Enabling physical entry, Physical theft	Accelerometer, Air quality, Air pressure, Microphone	[6, 29]
Window shade	Espionage, Voyeurism	Accelerometer, Power, Microphone, Light	[13, 152]
Light	Data exfiltration, Espionage, Causing physical harm	Light, Power	[51, 162]
Smart cam	Espionage, Voyeurism	Power, Accelerometer	[201]
Thermostat	Enabling physical entry, Extortion	Temperature	[6, 49]
Door lock	Enabling physical entry, Physical theft	Accelerometer, Microphone	[50]
Siren	Causing physical harm, Misinformation Causing environment damage,	Light, Microphone	[17]
Kettle, Oven, Iron	Causing physical harm, Causing environment damage	Power, Temperature, Humidity, Accelerometer, Microphone	[22, 29]

are systems in which events (*e.g.*, device state changes or measurement updates) reported by one device trigger actions (*e.g.*, unlocking the door or changing the room temperature) by other devices [160]. These systems enable users to remotely control and monitor devices and automate everyday tasks.

Unfortunately, as the focus in the development of smart home devices is often on the functional aspects, security is usually not the highest priority [1, 49]. In the past, this has opened up devices to attacks and infection by malware. The nature of IoT devices permeating a user’s life thus comes with real dangers. Sensitive data might be leaked, actuators could enable physical interaction within peoples’ houses, and distributed attacks may be launched using the myriads of installed smart home devices.

There has been a significant amount of work dedicated to reducing the risk posed by insecure IoT devices. Existing work has focused on either device and smart application permission systems [6–8], or network traffic analysis [9, 10, 43]. But none of these approaches can protect event integrity after a device has been compromised.

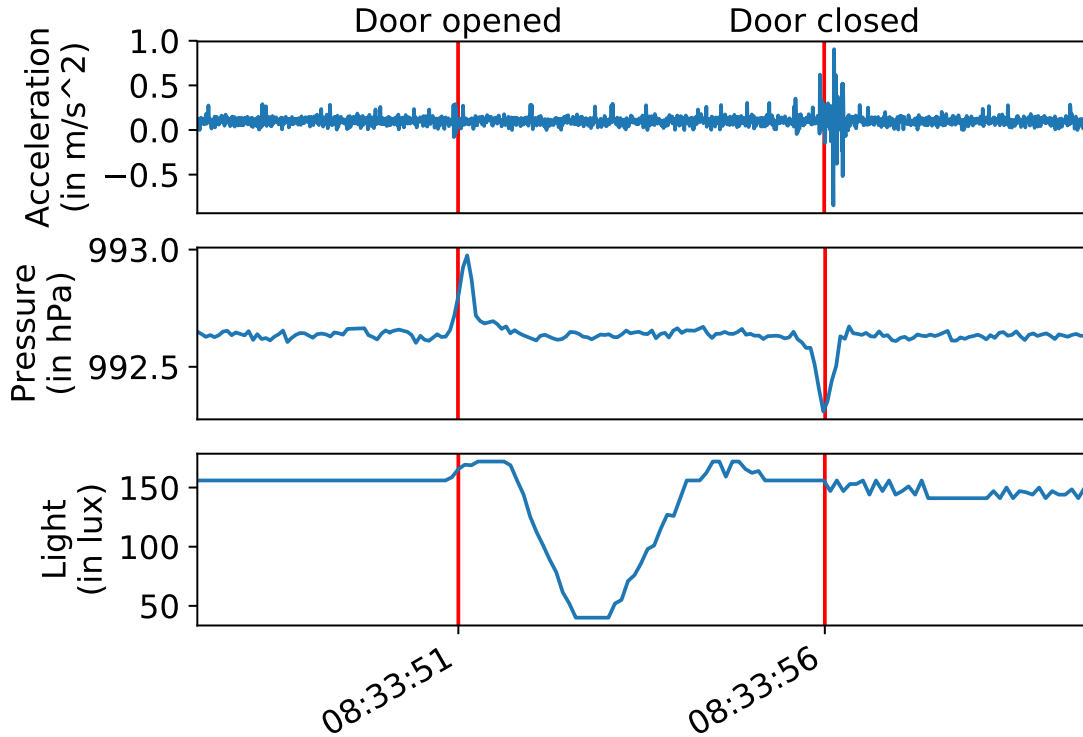


Figure 5.1: An example of how door events can affect different sensors. The red lines correspond to the timing of first a door opening event and then a door closing event. One can see that the door closing event has a clear effect on all three sensors, whereas the door opening event affects primarily the light and pressure sensors.

A compromised device can control what data it sends to the smart home system and thus lie about what it is really doing or sensing. Possible attacks range from smart cameras being turned on without the knowledge of the occupants, to smart locks not locking the door as instructed. An interesting property of smart home systems is the interconnectedness of events. A device that seems to have no security relevance can become an attack vector if it triggers actions by other devices (*e.g.*, a system that unlocks the back door when the kitchen light is being turned on [22]). An overview of smart home devices and potential attacker goals can be found in Table 5.1.

Despite the risk posed by the presence of cheap and often insecure IoT devices, their pervasiveness also opens up new ways to strengthen the security of smart home systems, as these devices are often equipped with a variety of sensors whose readings allow us to draw conclusions about their environment. Because these sensors can measure the physical effects of device actions and events, they can thus

be used as a secondary source to verify that these events have actually occurred. As an example, Figure 5.1 shows how a closing door affects an air pressure sensor, a light sensor and an accelerometer placed within the same room. Table 5.1 provides a (non-exhaustive) overview of which sensors are expected to sense effects of events relating to current smart home devices.

In this chapter, we introduce PEEVES, a system that automatically verifies events in smart homes based on physical event signatures. As events are fed into the system, it uses the distributed sensor data for supervised learning of the corresponding event signatures. By leveraging sensors available in the same physical environment, PEEVES can verify if reported events have actually occurred. Our system is able to automatically learn these signatures and determine which sensors are relevant for verification—without supervision by the user. We demonstrate the feasibility of PEEVES in a real-world experiment and evaluate it against a zero-effort attacker as well as two advanced types of attackers. The zero-effort attacker spoofs or masks events at arbitrary times. Therefore, their success hinges on the timing of their attack coinciding with sensor anomalies that result in the system misclassifying the event in question. The first advanced attacker introduces the notion of an opportunistic attacker who can wait for and predict the right attack opportunity in order to increase the impact of their attack. The second type of advanced adversary considered in this chapter is an attacker who has the ability to compromise parts of the system during operation and can thus create attack opportunities of their own volition.

In this chapter, we make the following contributions:

- We introduce PEEVES, a system to automatically learn smart home event signatures for various sensing modalities (**RQ3**).
- We test PEEVES in a real-world experiment with 48 sensors and 22 different event types over two weeks.

- We use the resulting dataset to evaluate PEEVES against spoofing and masking attacks, and we make this dataset publicly available.¹
- We introduce a method for the attacker to predict attack opportunities in order to increase their success rate and show that PEEVES is still effective against this strong adversary (**RQ4**).
- We evaluate PEEVES against an adversary that can compromise parts of the verification system and demonstrate how sensor fusion can be used to improve resiliency against this kind of attack (**RQ4**).

The remainder of the chapter is organized as follows: Section 5.2 gives an overview of smart home background. Sections 5.3 and 5.4 outline our system and threat model. We describe our experimental design in Section 5.5 and our methods in Section 5.6. We present our results in Section 5.7, discuss real-world issues and limitations in Section 5.8, and conclude the chapter in Section 5.9.

5.2 Background

5.2.1 Smart Home Systems

There are many competing smart home systems ranging from commercial solutions such as Samsung’s SmartThings [161] and Apple’s HomeKit [202] to open-source tools like OpenHAB [203] and HomeAssistant [204].

Even though these systems use different terminology, they all have the same fundamental conceptual design. Each system uses a smart hub as a central node that aggregates data and controls the interactions between devices. Additionally, the hub often connects devices to external cloud services. Devices can have sensors measuring their physical environment, as well as actuators that interact with it. For example, the opening and closing of a door can be detected by a magnetic contact switch (*sensor*) mounted on the doorframe, whereas a smart light switch (*actuator*) can be used to remotely control a ceiling light.

¹<https://www.doi.org/10.5287/bodleian:mv22Jk2Xj>

These devices have internal states that are representations of the actual physical states they are monitoring. In the previous example, the door can be either “open” or “closed”, while the ceiling light has the states “on” or “off”. *Physical events* are state changes that can be triggered by the user directly (*e.g.*, opening a door), through a user interface (*e.g.*, “Alexa, turn on the kitchen light”), or automatically by the system hub (*e.g.*, lower blinds at sunset every day). When a smart device notices that a monitored physical state changes, the device notifies the hub by sending an *event notification*. Note the difference between physical events and event notifications. Physical events are the phenomena happening in the real world, whereas event notifications represent the view that a smart device has (or claims to have) of those phenomena.

Automations of devices are configured by the user through rules that typically follow the trigger-action principle. For example, a rule can state that whenever the kitchen door opens (*trigger*), the ceiling light in the kitchen should be turned on (*action*). In many systems, several actions can be grouped together by creating *modes* to simplify rules. If the user wants to open the blinds and make a cup of coffee every time their alarm goes off, they can create a “good-morning” mode that triggers all these actions at a certain time rather than program them all individually.

5.3 System Model

We consider a system model inspired by contemporary smart home systems as described in Section 5.2. These systems are controlled by a central hub, and all their traffic is routed through that hub.

Besides the hub, PEEVES consists of several heterogeneous smart home devices. Devices have attributes that describe what they can do. These attributes can be possible device states or measurements, and supported actions.

In our system, we differentiate between two different attribute types. Attributes related to physical events are considered to be *event sources*. These attributes consist of a finite list of possible device states or actions. For example, a door

sensor has the states “open” and “closed”, whereas a window shade understands the actions “up” or “down”.

When the state of a device *changes* in the real world, we refer to it as a *physical event*. For example, this can be someone turning the door handle and pushing the door to open it, or the window shade moving from one position to another. We define *event notifications* as reported state changes occurring at time t . An event notification is represented by a tuple $E_t = (d, a)$, where d is the affected device and a is the changed attribute of the event source. We assume that events are reported in a timely manner, *i.e.*, the system rejects delayed event notifications. This is a parameter that should be chosen to accommodate the expected network delay (*e.g.*, $\sim 20\text{-}200\text{ms}$ for Zigbee with a maximal payload size of 150 bytes and a maximal hop size of 6 [205]). We disregard this parameter in our analysis for the sake of simplicity.

The goal of PEEVES is to verify if the physical event associated with a received event notification did in fact occur. Hence, the verification of a physical event is triggered by the reception of an event notification.

Other sensors of various datatypes provide measurements and are used as *verification sensors*. Data collected by verification sensors gets transmitted to the central hub for further processing. This data is then being used to learn the sensor-specific physical fingerprints of events. Event verification is done by the hub and is based on the aggregate data of all the sensors. We assume that event sources and verification sensors remain uncompromised during training and that the hub system software remains uncompromised during operation as well as training.

PEEVES provides a countermeasure to *event spoofing*, *i.e.*, when an event source or a malicious application running on the hub [17] reports an event that has not physically happened. The system can additionally be used to check for *event masking*, *i.e.*, when an event notification is suppressed for a real physical event. In this case, the system continuously checks for event occurrences on a rolling window of sensor data. An overview of PEEVES is given in Figure 5.2. PEEVES is designed to run autonomously in the background, alerting the user of potentially fraudulent events through a mobile phone app. The user is only required to provide feedback

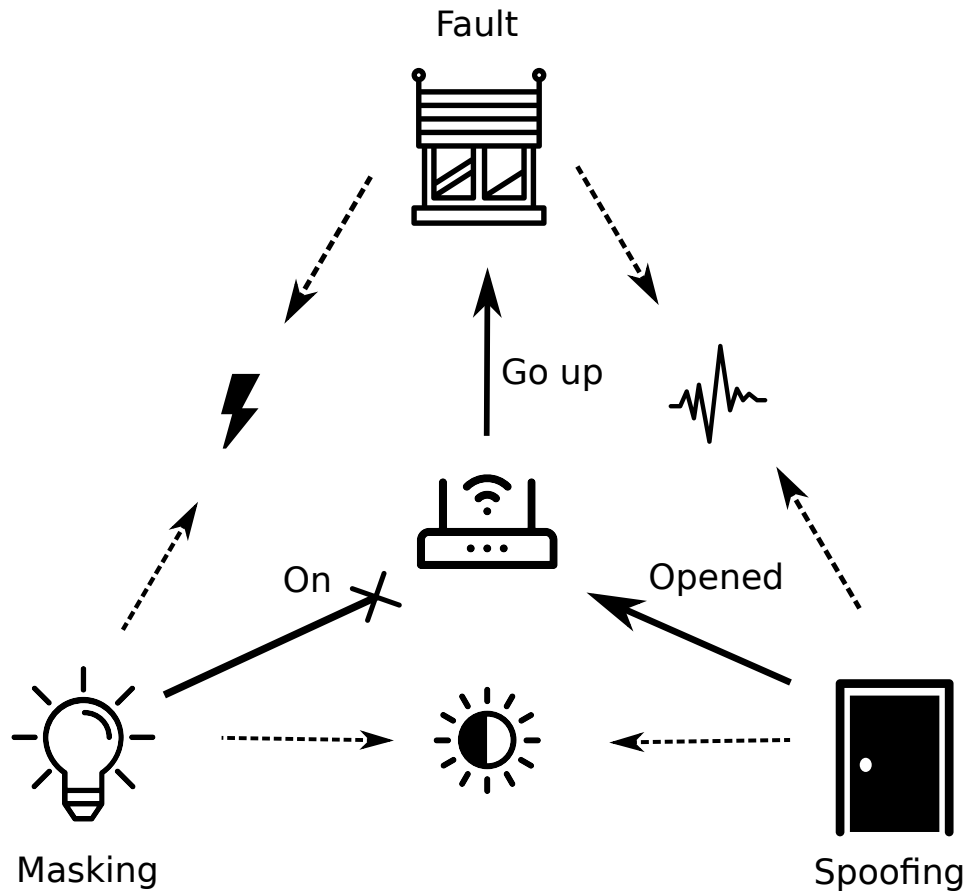


Figure 5.2: This figure gives an overview of PEEVES and three exemplary scenarios. The window shade at the top exhibits a fault and does not act on a command by the smart hub in the middle. On the bottom right, the door sensor sends a spoofed event to the hub, falsely claiming that the door has been opened. To the left of it, the light bulb has been turned on without sending an update to the hub—masking the event. PEEVES uses verification sensors to check the physical fingerprints of the corresponding events. This is possible as these events have measurable effects (dashed lines) on sensors measuring physical phenomena such as power meters (top left), accelerometers (top right) or light sensors (bottom).

to the system if an alert is raised, at which point they can advise the system how to proceed. For example, a user would dismiss an alert if an event has been marked as spoofed, but the user knows that the event in question has in fact occurred. Otherwise the user would instruct the system to suppress the illegitimate event.

5.4 Threat Model

We consider an attacker that is able to remotely compromise event sources in the smart home. This allows them to trigger any supported actions or events directly, bypassing the hub. The attacker is, however, not able to make any physical changes to the devices or events. Furthermore, the attacker is able to choose when and if they report an event to the hub. The attacker is thus able to launch both spoofing and masking attacks for the compromised event source. However, events that have a large delay between event timestamp and time of notification will be automatically rejected by the system (*i.e.*, the attacker can not delay their decision to launch the attack).

For example, an attacker might want to gain access to a house. Similar to the example in IoTGuard [22], the back door of the house gets unlocked when the smart home enters a “home mode” which is triggered when the front door opens. The attacker can then use a compromised door sensor to *spoof* an opening door event to trigger the “home mode” and thus unlock the back door. Similarly, an attacker might *mask* an event that warns the user that the privacy mode of a smart camera has been turned off if they want to spy on the inhabitants without alerting them.

However, mismatches between the internal state of a smart home system and reality are not limited to malicious behavior. In our experiments, we have experienced how device faults can have similar effects to attacks. Examples include a window shade not executing commands despite acknowledging them (*e.g.*, due to being stuck) and a smart radiator valve that did not notice that it was constantly heating due to a blocked actuation pin. These kinds of faults are particularly common with “retrofitted” smart devices, *i.e.*, regular devices with added smart control logic. The less integrated the smart and normal components of the device are, the more likely is a mismatch between the device’s reported and actual state changes.

Our system considers three types of attackers of increasing severity:

- *Zero-effort attacker/Sensor fault*
- *Opportunistic attacker*

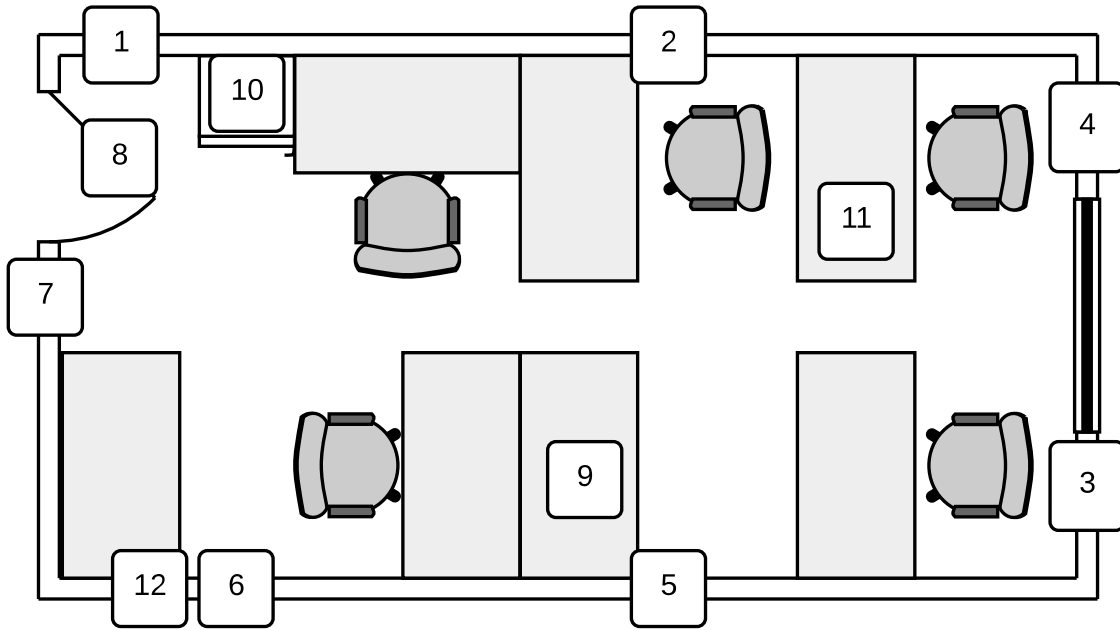


Figure 5.3: The device deployment for the experiment is pictured in this floorplan. The numbers refer to the group numbers in Tables 5.3 and 5.4. Sensor group 8 is mounted on the door, groups 9 and 11 are placed on tables, and group number 4 is fixed to a shelf. All the other sensors are mounted on the wall.

- *Sensor-compromise attacker*

For all attacker types, event sensors are considered to be untrusted, as they can be compromised or exhibit random faults. None of the attackers has physical access to the devices. The zero-effort attacker spoofs or masks events at arbitrary times and does not have knowledge of any system parameters such as the features used for detection. Furthermore, this attacker can not read the data of the verification sensors.

In comparison, the opportunistic attacker knows all the system parameters, such as the features used for detection, and has read access to the current data of all sensors. Using this information, this attacker tries to spoof or mask only at times when they think they will be likely to succeed. The attacker's goal is to maximize the probability of their spoofed or masked event going undetected (*i.e.*, being incorrectly judged as genuine) while minimizing the amount of time they have to wait for an attack opportunity. Neither the zero-effort nor the opportunistic attacker can modify or suppress data originating from verification sensors.

Finally, a sensor-compromise attacker is an extension of the opportunistic attacker. In addition to the above capabilities, a sensor-compromise attacker can also modify the sensor data of a subset of verification sensors in real-time (*e.g.*, replay a correct sensor signature of a compromised accelerometer for the “door closed” event). The attacker uses this to create opportunities by manipulating the compromised sensors’ data streams. The sensor-compromise is still bound by the system requirement of timely sensor data (*i.e.*, they can not retroactively modify sensor data).

5.5 Experimental Design

To evaluate the feasibility of our proposed system, we conduct an experiment in which we collect the physical signatures of 22 event types in a smart office environment. The experiment is conducted continuously over two weeks while the office is being used by four people for their everyday work.

5.5.1 Overview

During the study, we collect data for a diverse range of events and sensing modalities. In an office, we distribute thirteen Raspberry Pis, each equipped with several environmental sensors to collect data. The deployment for this experiment can be found in Figure 5.3, while a detailed sensor configuration is listed in Table 5.3. A description of the sensor types used in the experiment can be found in Table 5.2. We consider as event sources: an office door with automatic closing mechanism, a sliding window, a ceiling light, an automatic window shade, a fridge, an automated tower fan, a smart radiator valve, a smart coffee machine, a PC with attached screen and a smart camera with privacy mode. Over the whole two week duration of the study, we registered 2773 events in our smart environment. A description of the event sources together with their corresponding events and event occurrences can be found in Table 5.4.



(a) The sensor configuration for one of the Raspberry Pis. A detailed listing of the sensor configuration can be found in Table 5.3. The contact switch at the top is being used to detect when the window is being opened or closed.

(b) The fan can be remote controlled through a Raspberry Pi Zero acting as an IR remote. Light switch presses are recorded through an attached force-sensitive resistor.

(c) The user interface running on an Amazon Fire tablet. It allows users to control the fan, radiator and window shade, as well as make coffee.

Figure 5.4: Experimental setup

Table 5.2: The different sensor modality types considered as verification sensors in the real-world experiment. The number of sensors deployed of a certain type is given in parentheses.

Abbreviation	Sensor modality (#)
A	Accelerometer (10)
G	Gyroscope (10)
M	Magnetometer (1)
P	Air pressure (9)
H	Humidity (5)
T	Temperature (10)
L	Light (7)
PM	Power meter (7)
AQ	Air quality/ CO_2 (2)
S	Sound pressure level (6)
R	Received signal strength (6)
TC	Thermal camera (2)

Table 5.3: The sensor configuration for the real-world experiment. The sensor modalities covered by each sensor are given in parentheses. The used abbreviations can be found in Table 5.2. The deployment is pictured in Figure 5.3.

Group #	Device	Sensor (Modalities)
1	Pi #1	MPU6050 (A/G)
		BMP280 (P/T)
		TSL2560 (L)
		USB microphone (S)
		Ralink RT5370 USB WiFi dongle (R)
2	Pi #2	MPU6050 (A/G)
		BMP280 (P/T)
		TSL2560 (L)
		USB microphone (S)
		Ralink RT5370 USB WiFi dongle (R)
3	Pi #3	MPU6050 (A/G)
		BME680 (P/H/T)
		TSL2560 (L)
		SGP30 (AQ)
		USB microphone (S)
		Ralink RT5370 USB WiFi dongle (R)
4	Pi #4	MPU6050 (A/G)
	Plug #3	Contact sensor (window 1)
	Plug #4	TP-Link HS110 (PM of window shade)
5	Pi #5	TP-Link HS110 (PM of smart cam)
		MPU6050 (A/G)
		BMP280 (P/T)
		TSL2560 (L)
		USB microphone (S)
6	Pi #6	Ralink RT5370 USB WiFi dongle (R)
		MPU6050 (A/G)
		BMP280 (P/T)
		TSL2560 (L)
		USB microphone (S)
7	Pi #7	Ralink RT5370 USB WiFi dongle (R)
		Force-sensitive resistor (ligh switch)
		MPU6050 (A/G)
		BME680 (P/T/H)
		TSL2560 (L)
		SGP30 (AQ)
8	Pi #8	USB microphone (S)
		Ralink RT5370 USB WiFi dongle (R)
		ST Micro LPS25H (P/T)
		ST Micro HTS221 (H/T)
9	Pi #9	ST Micro LSM9DS1 (A/G/M)
		Contact sensor (door)
		Pi #13
		Fan remote control
		Plug #5
10	Pi #10	TP-Link HS110 (PM of fan)
		Plug #6
		TP-Link HS110 (PM of PC)
		Plug #7
		TP-Link HS110 (PM of screen)
11	Pi #11	MPU6050 (A/G on fridge)
		BME680 (A/T/H on fridge)
		BME680 (A/T/H inside fridge)
		TSL2560 (L inside fridge)
		Contact sensor (fridge)
12	Pi #12	Plug #1
		TP-Link HS110 (PM of fridge)
		Plug #2
		TP-Link HS110 (PM of coffee machine)
		MLX90640 (TC)
		MLX90640 (TC)

Table 5.4: This table shows the event sources considered in the real-world experiment and their corresponding event notifications. The number of event occurrences is given in parentheses. The group number refers to the deployment group number pictured in Figure 5.3.

Group #	Event source	Event notifications (#)
3	Window	Window opened (44) Window closed (44)
	equiva eQ3 radiator thermostat	Radiator on (41) Radiator off (41)
4	Samsung SmartCam SNH-VP6410PN	Camera on (156) Camera off (156)
	Teptron Move window shade controller	Shade up (74) Shade down (75)
7	Light switch	Light on (47) Light off (48)
8	Door	Door opened (340) Door closed (340)
	Ring doorbell	Doorbell used (52)
9	Desktop PC	PC on (73) PC off (72)
	LCD screen	Screen on (144) Screen off (213)
	Tower fan	Fan on (334) Fan off (335)
10	Mini fridge	Fridge opened (56) Fridge closed (56)
	Smarter Coffee	Coffee machine used (34)

5.5.2 Data Collection

The ground truth for the events is collected as follows. The door, window and fridge are outfitted with magnetic contact sensors, shown in Figure 5.4(a). Light switch events are registered through a force-sensitive resistor that registers when the switch is being pressed, see Figure 5.4(b). The smart cam follows a schedule, periodically switching between privacy mode and normal mode every hour. When the smart cam is in privacy mode, its camera is tilted towards its case, preventing

Table 5.5: This table gives an overview of which events were triggered manually or automatically. For automatically triggered events, the schedule of device actuations is given.

Event source	Manual	Automatic	Schedule
Camera	✗	✓	Toggle every hour (24/7)
Coffee machine	✓	✗	—
Door	✓	✗	—
Doorbell	✓	✗	—
Fan	✓	✓	Night: 2 min every 30 min day: 2 min every 2 h
Fridge	✓	✗	—
Light switch	✓	✗	—
PC	✗	✓	2 h on, 2 h off (24/7)
Radiator	✓	✓	Twice per day for 30 min once for 15 min
Screen	✗	✓	30 min on, 30 min off during PC duty cycle
Window shade	✓	✓	6-9am: toggle every 30min 7-9pm: toggle every 30min
Window	✓	✗	—

it from recording video from the room. Window shade, radiator, fan and coffee machine can be controlled via a tablet, see Figure 5.4(c). As the fan used in the experiment does not have any network connectivity, we automate it by using a Raspberry Pi Zero as an infrared remote control, shown in Figure 5.4(b). The doorbell events are being polled from the Ring servers.

Events are triggered sporadically by the office occupants and automatically at certain times. Smart cam, PC and screen events are only triggered automatically, whereas door, window, fridge, doorbell, and coffee machine events are only triggered manually. Automatic events follow a schedule with randomized starting times to avoid systematic cross-contamination of event signatures. More details on event schedules are given in Table 5.5.

We use the Network Time Protocol (NTP) to synchronize time across the

Raspberry Pis used in the experiment. The sensors connected to the Raspberry Pis are continuously being polled over their I^2C interface. The measured data is being saved locally on USB sticks. As an example, the sensor configuration of one Raspberry Pi is shown in Figure 5.4(a).

In addition to the sensors connected to the Raspberry Pis, we use TP-Link HS110 power meters to measure the voltage, current and power usage of the corded devices. The measurement data of the power meters is continuously polled over WiFi.

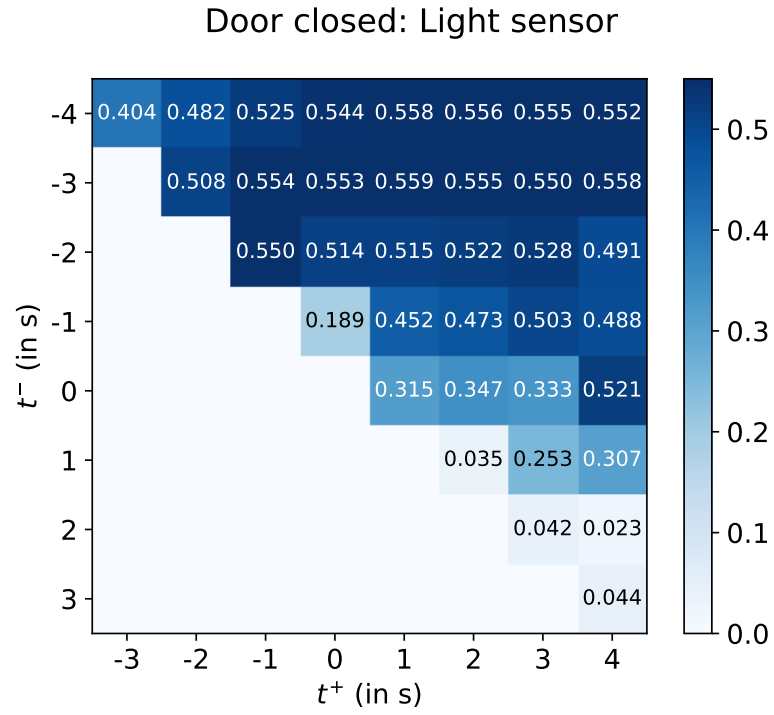
5.5.3 Ethical Concerns

This project has been reviewed by and received clearance by the responsible research ethics committee at our university (approval number CS_C1A_19_009). The main concerns relate to the door contact sensors, thermal cameras and the microphones used as sound pressure sensors. The door contact sensors could, in theory, be used to track people's movement (*i.e.*, when they enter or leave the office). However, as the office is used by multiple people, it would not be possible to relate this data to individuals. In addition, individuals can enter or leave the room freely if the door is propped open. Naturally, microphone data would be far more sensitive. To avoid this issue, we use the microphones only as sound pressure sensors that record sound levels, rather than the raw audio. As such, it is impossible for sensitive audio data (*e.g.*, conversations) to be recorded. Raw sound pressure values allow the detection of noise, but they do not allow one to accurately determine the event or activity that caused it. Despite their low resolution, the thermal cameras could be used as low-quality video cameras, as the outlines of humans can be easily distinguished from the colder environment. Therefore, we only collect column and row averages of thermal image frames and thus make it impossible to recreate the original video data.

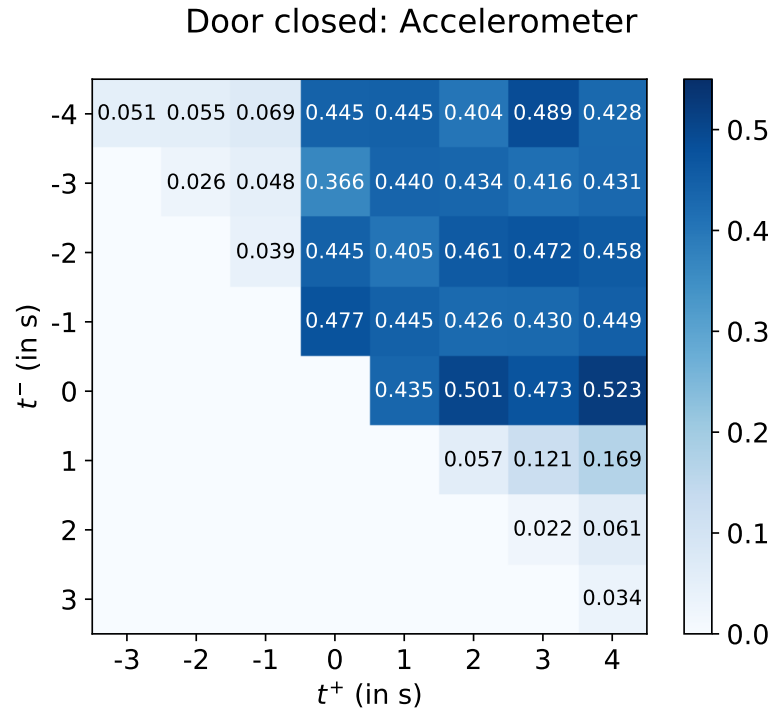
5.6 Methods

5.6.1 Data Labeling

Separately for each event, we label all points in time with either 0 (no event occurred) or 1 (event occurred). The timestamps of 1-samples are identified by the



(a) Light sensor



(b) Accelerometer

Figure 5.5: Window parameter grid search for the “Door closed” event. Values given are the average RMI.

corresponding ground truth (*e.g.*, the precise moment a door closes as identified by the contact switch). The event time for automatically triggered events is the moment the command is sent to the device. 0-events are generated periodically in 1-second intervals for the remainder of time. For the spoofing attack detection this approach initially leads to “ambiguous” labels close to events. For example, if a door closes at time t , the system will likely detect (and correctly verify) this in a small window around t . The fact that a 0-sample very close to a 1-sample is classified as 1 (*i.e.*, potential spoofing opportunity) does arguably not pose a problem. While this would, in theory, allow the attacker to spoof events just before or after actual events, there would be no incentive to do so. In order to deal with this, we use so-called safety margins around events. We remove samples measured during such a safety margin from both the training and test data. We set the size of the safety margin to twice the size of the largest sensor window. For the masking attack detection, however, we can not make use of the 1-samples with the exact time of the event as the event masking would make the actual timestamp unknown to the system. Instead, rather than removing the 0-samples within the safety margin around the event, we label these samples as 1-samples for the sake of evaluation, as these are the samples for which the rolling window overlaps with the sensor signature. Therefore, a masking attack is successfully detected if any of these 1-samples is classified as such.

5.6.2 Dataset Division

We split our 2-week dataset into three parts: development, training and testing. Each of these parts forms a block of consecutive samples, rather than choosing the appropriate fraction of samples randomly over the entire dataset. This reflects actual system operation, for which the training phase has to predate testing in its entirety. This is especially crucial as we require the event sources to remain uncompromised during the training phase. We use the first day of data for development and split the remaining data evenly into training and testing sets.

The development set is used for two purposes: Calculation of the sensor-specific windows and feature selection. We describe the methods for both of these in the

following subsections. Based on the selected features, we then train and test the classifier on the remaining two parts.

5.6.3 Sensor Window Selection

Each event is associated with a single timestamp (*e.g.*, the precise moment the contact sensors detects a closing or opening door). However, the physical signature of the event can precede, follow or overlap with the event. An illustration of these differences is shown in Figure 5.1. Even for the same event, the time window in which the event signature is noticeable depends on the sensor. For example, the door closing is preceded by changes in light and followed by an accelerometer response. This event signature window defines the period over which sensor data is collected to verify the event (see the next subsection for details on feature extraction). While it would be possible in principle to manually set these sensor windows for each event based on the data, we develop an automatic approach that is used before the system’s training phase:

The window size and position relative to the event is controlled by two parameters: t^+ and t^- . Given an event timestamp t , the corresponding window will be the interval $(t - t^-, t + t^+)$. For each event, we obtain the optimal values for both parameters through a grid search on the development set. The performance indicator for this grid search is the average relative mutual information (RMI, see next subsection) of all features. In order to avoid distortion of the average by very poor features, we exclude features that do not perform better than a “random” feature. An illustration of the results of this grid search for the same event and two different sensors can be seen in Figure 5.5. It is evident that the window for the light sensor largely precedes the event (*i.e.*, negative values of t^-) whereas the accelerometer response achieves the best distinctiveness in a 4-second window directly following the event.

5.6.4 Feature Extraction

The baseline values of many of our environmental sensors fluctuate heavily throughout the day (*e.g.*, the air pressure readings are dependent on the ambient air

pressure outside). As we still want to capture transient changes caused by events, we have to compute changes relative to the current baseline. To do so, we compute the features in the following manner. For each 0-sample and each 1-sample (see Section 5.6.1), we construct the event signature window as described in the previous section using parameters t^+ and t^- . As the current baseline, we subtract the mean of the preceding window with the same length as the event signature window. We then compute five statistical features (min, max, mean, sum, and standard deviation) on the corresponding event signature window on the data of every sensor.

Following the feature extraction, we measure each feature’s quality (*i.e.*, distinctiveness). This is needed for the computation of ideal window sizes for each sensor (see following subsection) and feature selection. We use relative mutual information (RMI) as a quality measure. RMI is defined as

$$\text{RMI}(\text{event}, F) = \frac{H(\text{event}) - H(\text{event}|F)}{H(\text{event})}$$

where $H(A)$ is the entropy of A and $H(A|B)$ denotes the entropy of A conditioned on B . *event* denotes the ground truth for this sample (*i.e.*, 0 or 1) and is therefore discrete. However, the feature space for most features is continuous. In order to precisely determine the conditional entropy, binning would be required to discretize the features. However, the reported RMI would depend on the binning strategy and number of bins (with more bins leading to a higher calculated RMI). To avoid this problem, we use the non-parametric approach proposed by Ross *et al.* to estimate the mutual information between the discrete event ground truth and continuous features [206].

Due to the small number of 1-samples and the resulting class imbalance, features often show a comparatively high RMI in the development set without actually providing systematic information about the event. This is particularly critical as feature selection is performed on a small subsample of the entire data. Due to timed automated events, events sometimes occur at the same time. While this is not systematic, even a small number of overlaps can lead to overstated RMI.

In order to limit the effect of noisy features and reduce model training time, we select features with an RMI above 40%.

5.6.5 Classification and Metrics

In this chapter, we compare two decision making processes. The first approach uses a monolithic classifier, whereas the second approach uses an ensemble of classifiers trained on individual sensors to provide more robustness when faced with compromised sensors. For both approaches, we use a binary linear support vector machine (SVM) as our classifier. We choose a linear kernel due to the large amount of training data (roughly 600,000 samples) and high number of features. The classifier is parametrized by the penalty parameter C . Since the two classes (events and non-events) are highly imbalanced, we assign different values of C proportional to the fraction of samples for each class. This avoids biasing the classifier towards the more common class (*i.e.*, non-events).

Monolithic Classifier

For the monolithic classifier approach, we train a single SVM classifier on all the features extracted through the process described in Section 5.6.4. Therefore, all the sensors with features that have sufficiently high RMI values could possibly influence the decision of this classifier. Instead of using the actual binary classifier decision, we calculate each sample's distance to the decision boundary. This allows us to then define a threshold on this distance to control the tradeoff between detection rate (DR) and false alarm rate (FAR). In a spoofing attack, the attacker flips a 0-sample (non-event) to a 1-sample (event). The DR is therefore defined as the fraction of 0-samples that are correctly classified (*i.e.*, the probability of detecting a spoofing attack conducted at a random time). Conversely, the false alarm rate is defined as the fraction of wrongly classified 1-samples (*i.e.*, actual events that are wrongly classified as spoofed). For a masking attack, the attacker attempts to do the opposite (change a 1-sample to a 0-sample). The DR is therefore defined as the fraction of correctly classified event periods (*i.e.*, the probability of detecting

a masking attack of an actual event in a time period equal to the safety margin around the event), whereas the false alarm rate is defined as the fraction of wrongly classified 0-samples (*i.e.*, the probability of wrongly classifying the occurrence of a masked event at a random point in time). When detecting masking attacks, we aggregate false alarms within a time period equal to the safety margin to emulate normal system behavior (*i.e.*, if several consecutive 0-samples are wrongly classified within this time period only the first one raises an alarm). We do this because false alarms typically occur in bursts due to the overlapping nature of the rolling windows of sensor data. We do not aggregate consecutive misclassified 0-samples for spoofing, as each of these offers a separate opportunity for the attacker. We combine both metrics into the equal error rate (EER), which is the error rate at a threshold where ($FAR = 1 - DR$). Since the relatively limited number of 1-events leads to an equally limited number of possible FARs or DRs for spoofing and masking, respectively, we use geometric interpolation (see Figure 5.8) to obtain the EER.

Sensor Fusion

As a more robust defense against compromised sensors, we use a fusion-based approach that takes the decisions of classifiers trained on the features of a single sensor and combines those decisions using a fusion rule. These single-sensor classifiers use the same feature extraction methods and are trained in the same way as the classifier trained on the features of all selected sensors. A sensor is selected for the sensor fusion if at least one of its features is selected by the feature extraction process detailed in Section 5.6.4.

We use a k-out-of-n fusion rule which decides that an event has occurred (is labeled as 1) when at least k out of the n sensors used for the fusion agree that it has occurred [207]. We choose this fusion rule on account of its recognized strong performance and ease of use [208]. As this fusion rule assigns equal weights to all classifiers regardless of individual accuracy, it is harder for the attacker to trick the system by compromising one important sensor. Additionally, as opposed to a

simple majority rule, the parameter k can be freely chosen to control the tradeoff between DR and FAR in addition to tuning the threshold of individual classifiers.

5.6.6 Attacker Implementation

As outlined in our threat model (Section 5.4), we consider three kinds of attackers: zero-effort attackers, opportunistic attackers, and sensor-compromise attackers. The zero-effort attacker chooses their moment for event spoofing or masking randomly without consideration for the physical environment. Conversely, the opportunistic attacker uses information about the verification sensors to spoof at times when they are particularly likely to be successful. Finally, the sensor-compromise attacker is an extension of the opportunistic attacker that can additionally compromise individual verification sensors.

Opportunistic Attacker

The goal of the opportunistic attacker is to predict whether a given point in time will be misclassified by the verification system (*i.e.*, at what time a spoofed event will be incorrectly classified as genuine or when a masked event will remain unnoticed). Intuitively, the attacker could copy the verification system since they have access to sensor readings and ground truth. Based on this system, they will learn the system's score for each potential point in time. However, this will only grant them posterior knowledge about opportunities (*i.e.*, they will learn that a good spoofing or masking opportunity would have been one second ago). In order to judge whether a time t is suitable for spoofing or masking, the attacker can only use information collected up to time t . Please refer to Figure 5.6 for a visualization of this restriction. To reflect this limitation, we build an attacker *surrogate model* as follows:

The attacker first uses the development dataset to calculate the ideal windows for each sensor. While the development set is relatively small (1 day), this is necessary to ensure that the attack is evaluated on the same size training and test set as the original system. In line with the attacker's limitations, they restrict the t^+ parameter to non-positive values. Intuitively, this means that only sensor

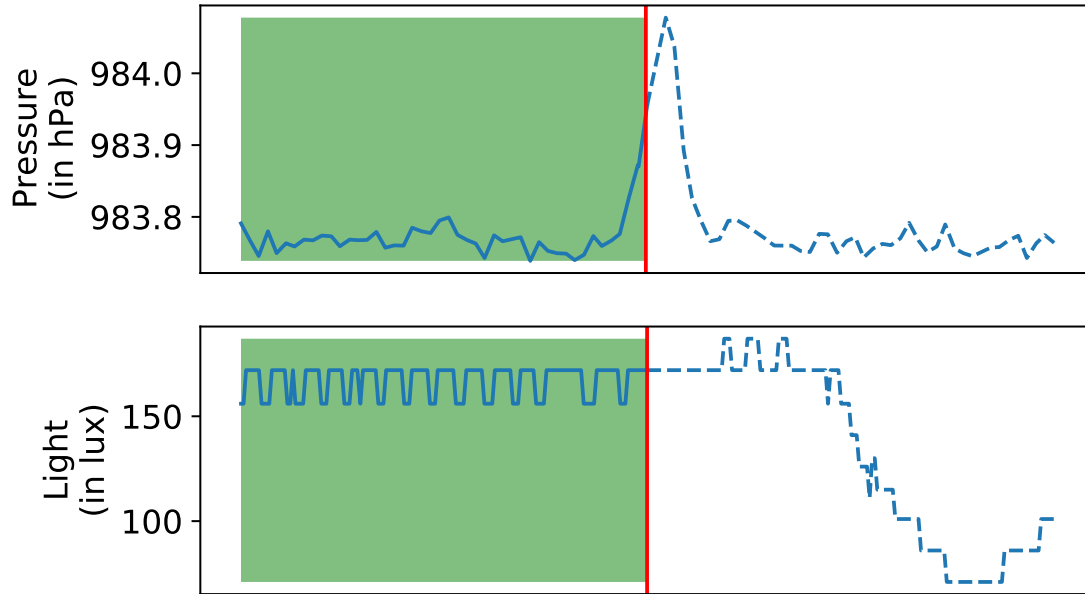


Figure 5.6: Door opened event as recorded by an air pressure sensor and a light sensor. In this example we assume that there is an attack opportunity that perfectly matches an actual event. The red vertical line indicates the time t of the event (or the targeted attack time), whereas the green shaded area shows the time window that the opportunistic attacker can use to predict if the attack can be successful. One can see that if the attacker has access to the air pressure sensor data, they might be able to predict that this is indeed a good attack opportunity. The data from the light sensor, on the other hand, is not able to provide any useful information for this decision, as the event signature is only present after the event takes place.

information collected before the “event” is used. They then follow the same methods as the actual system (*i.e.*, feature selection and classifier training). Ideally, the scores produced by the surrogate model will be correlated with those of the actual system. The attacker can then continuously classify samples when no events are happening to find an opportunity (*i.e.*, a sample that scores well on their model). The attacker can then define a threshold for an attack. This threshold controls the tradeoff between success rate (*i.e.*, the probability the system will judge the attacker’s spoofing as a genuine event or miss that a masking attack has occurred) and the waiting time until an opportunity is found.

Sensor-compromise Attacker

Recall that in addition to having read access to sensor data like the opportunistic attacker, the sensor-compromise attacker can change the data of a subset of

compromised sensors. The goal of the sensor-compromise attacker is to replace the data of the sensors under their control with data of legitimate looking event signatures in case of a spoofing event or data from a period of inactivity in case of a masking event. The attacker can replay previously recorded data of actual events or compute an average event signature to choose this replacement data. However, due to the real-time requirement for sensor data imposed by the system, they can not choose sensor data depending on data generated by uncompromised sensors.

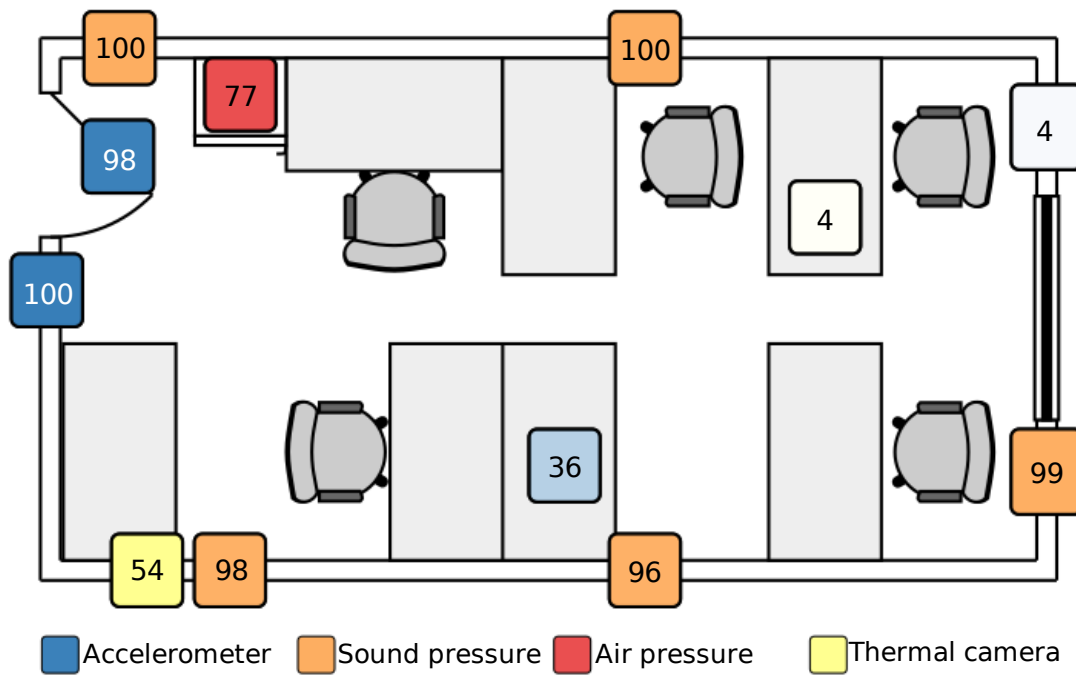
To show the potential of an attacker that is able to execute such an attack, we implement them as an attacker that has the ability to replace the relevant features before they enter the classifier to make a decision for the combined SVM and modify the decision of the affected single-sensor classifiers before they are aggregated by the fusion rule. We do this since we intend to capture and test the potential impact of the manipulation rather than a specific way of creating these signatures, and because we do not want to weaken the attacker by the choice of the attack signature. To attack the combined SVM, the attacker uses the development set to train a surrogate classifier on the data of each compromised sensor separately. The attacker then finds the sensor's sample in the development set with the highest score and uses this sample to replace the corresponding features before the features are passed to the system classifier.

If the attacker is also trying to wait for attack opportunities, the attacker additionally replaces the sensor data in the same manner before feeding it into the surrogate classifier used to predict attack opportunities.

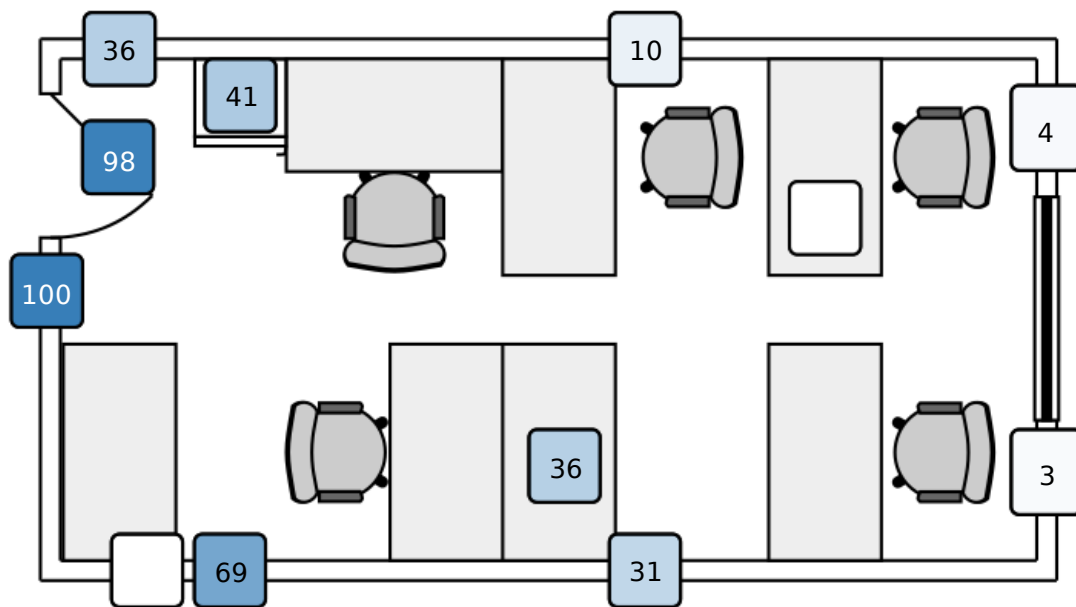
5.7 Results

5.7.1 Feature Distinctiveness

Table 5.6 shows the distinctiveness of feature groups for each event. Each cell shows the maximum RMI within the sensor group. Most results are relatively intuitive: Accelerometers detect door events, window shade events and the use of the coffee machine. Power meters primarily detect the devices they are connected to. Light sensors detect both toggling of the light switch and events that lead to obscuring the



(a) Most distinctive sensor based on their position. The number given for each sensor is their maximum RMI in %.



(b) Relative importance of accelerometer sensors depending on their position. Numbers are maximum RMI in %.

Figure 5.7: Type and quality of most distinctive sensor depending on its position

Table 5.6: Maximum RMI for each event within the different sensor groups. Sensor group labels are given in Table 5.2.

Event	A	G	M	P	H	T	L	PM	AQ	S	R	TC
Coffee machine used	87	67	4	12	11	24	82	100	75	37	9	37
Doorbell used	43	50	45	30	25	33	65	37	70	38	74	24
Door closed	99	96	96	88	41	69	99	35	41	100	66	53
Door opened	85	93	96	75	36	61	86	41	57	80	55	17
Fan off	24	11	2	2	2	53	9	100	12	62	27	11
Fan on	25	5	2	2	2	21	7	100	42	47	41	10
Fridge closed	87	77	5	98	54	92	99	45	33	73	61	67
Fridge opened	64	67	3	21	47	96	99	49	38	66	57	67
Light off	55	47	40	36	7	25	100	55	26	79	65	28
Light on	51	56	51	41	6	31	100	57	24	32	62	47
PC off	10	7	3	6	3	21	6	99	24	6	77	9
PC on	15	11	5	6	5	30	7	100	49	8	6	6
Camera off	7	5	3	4	4	7	18	99	17	5	40	5
Camera on	4	4	6	3	2	9	16	99	18	5	40	7
Radiator off	10	24	5	14	5	41	25	57	10	21	19	16
Radiator on	12	8	8	9	7	32	15	76	4	12	14	79
Screen off	6	3	4	5	3	14	22	99	15	4	38	4
Screen on	6	4	2	3	6	15	17	100	18	8	44	7
Window shade down	90	79	5	11	7	23	12	93	51	24	12	40
Window shade up	81	65	6	4	10	22	17	100	53	23	55	51
Window closed	16	20	11	19	5	12	91	64	35	75	95	34
Window opened	9	32	11	45	4	11	96	65	41	99	83	36

sensor (*e.g.*, a door opening in the proximity of the sensor). Signal strength (RSS) changes are particularly useful to detect window events. This is a consequence of windows only being operated manually. To do so, the user obscures the line of sight (LOS) between the sensor and the router, which causes predictable RSS changes. RSS changes are not only caused by LOS breaks, but also interference which makes RSS useful to detect the PC's power state changes. Note, that RMI is used to determine the distinctiveness of each individual sensor. Hence, the system is not dependent on a specific sensor configuration or deployment, but is instead able to learn relevant sensors *in situ*.

One of the more surprising results is the high distinctiveness of power meter features for virtually all events. Naturally, this effect is expected for devices that are attached to individual power meters (*i.e.*, fan, fridge, PC, screen, window shade, camera). With the exception of the fridge, the power meters pick up the spike in power once the device is turned on. When the fridge is opened and the temperature rises, the increased cooling leads to a (delayed) increase in power consumption. We observed a voltage spike in all power meters that occurs when the light is turned on or off which explains the relatively high RMI for these two events. This effect is particularly pronounced when the overall electrical activity is low. These results show that power meters are useful even beyond detecting state changes of their respective devices.

Figure 5.7(a) shows how different sensors contribute to detecting the door closed event. Sound pressure sensors achieve very high RMI that is independent from the sensor’s position. Interestingly, the air pressure sensor *inside* the fridge achieves a similarly high RMI. Our inspection of the raw signal suggests that the door closing actually causes a pressure change inside the fridge and that the reading is not a technical artifact (such as vibration).

Figure 5.7(b) shows the importance of accelerometers in detecting the door closed event depending on the position of the individual sensor. This importance is not purely down to distance from the door. In fact, sensors on the lock side of the door show higher RMI despite being further away. Unlike sound pressure, the accelerometer signal goes virtually undetected at the other end of the room.

5.7.2 Spoofing Attack Detection

The classification results for the spoofing attack detection are shown in Table 5.7. For each event, we list its EER, the detection rate at a threshold that does not cause false alarms and the false alarm rate at 99% detection rate. For 9 out of 22 events, we achieve perfect classification (*i.e.*, an EER of 0% or a 100% detection rate at 0% FAR). Some of these events (*e.g.*, door events) draw this good performance from being detected by a multitude of sensors. Others (such as light events) have few,

Table 5.7: Error rates for the spoofing attack detection of different events given in percentages. Numbers in brackets signify the absolute number of undetected events.

Event	#Events	EER	DR(FAR=0)	FAR(DR=99%)
Coffee Machine used	20	0.00	100.00	0.00 (0)
Door closed	177	0.00	100.00	0.00 (0)
Door opened	179	0.00	100.00	0.00 (0)
Fan on	165	0.00	100.00	0.00 (0)
Fridge closed	29	0.00	100.00	0.00 (0)
Fridge opened	28	0.00	100.00	0.00 (0)
Light off	26	0.00	100.00	0.00 (0)
PC on	33	0.00	100.00	0.00 (0)
Screen on	70	0.00	100.00	0.00 (0)
Camera on	70	0.01	99.99	0.00 (0)
Light on	25	0.02	99.98	0.00 (0)
Camera off	69	0.10	99.90	0.00 (0)
Fan off	162	0.17	99.82	0.00 (0)
Screen off	105	0.29	99.71	0.00 (0)
PC off	35	0.39	99.61	0.00 (0)
Radiator on	19	0.97	98.93	5.26 (1)
Window opened	24	4.01	0.35	4.17 (1)
Window shade up	33	11.39	0.28	15.15 (5)
Window shade down	32	12.04	0.19	12.50 (4)
Doorbell used	27	15.06	0.36	55.56 (15)
Window closed	24	15.24	17.63	33.33 (8)
Radiator off	21	49.41	7.65	100.00 (21)

but extremely reliable features. In order to more closely investigate events with less ideal performance (such as window events), we show their Receiver Operating Characteristics (ROC) curve in Figure 5.8. The ROC shows the tradeoff between DR and FAR. For most events shown in the curve it is evident that few instances of the event were assigned a very poor classification score. If the threshold is adjusted to (incorrectly) reject them as spoofs, the detection rate almost instantly jumps to 100%. This is most likely due to the event being triggered in a way that is very different from the training data (*e.g.*, a person opening a window only an inch). This is a natural consequence of our largely uncontrolled experimental design.

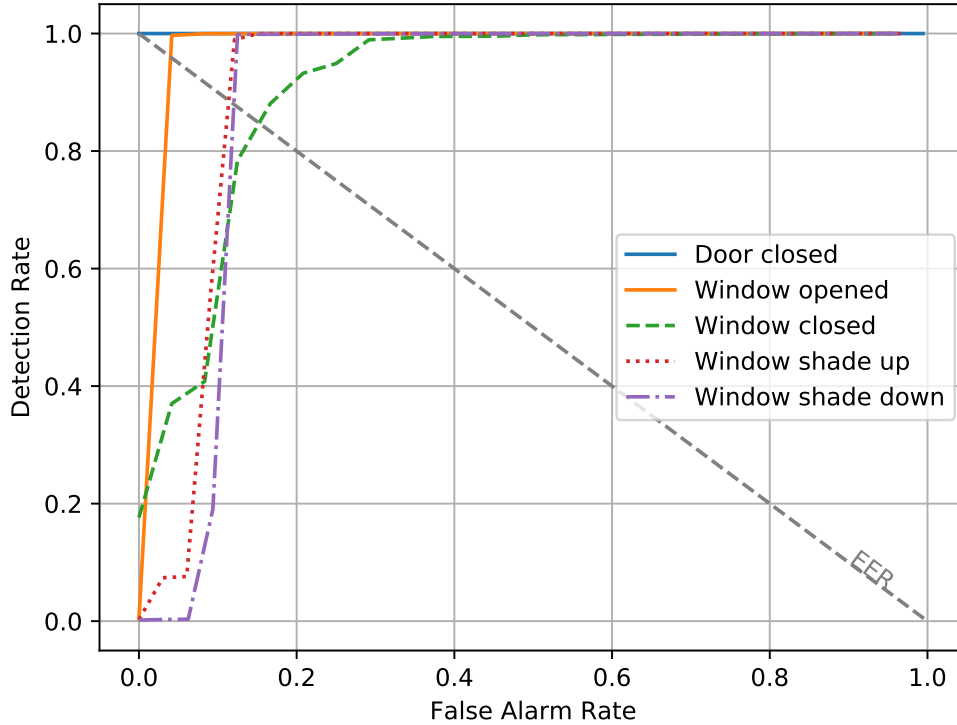


Figure 5.8: Receiver Operating Characteristics (ROC) curve for the spoofing attack detection of different events. Here we only show events with an EER significantly higher than 0 (well-classified events with an EER of 0 are a flat line at the top).

5.7.3 Masking Attack Detection

The classification results for the masking attack detection are shown in Table 5.8. As for the spoofing attack detection, we list the EER, the detection rate at a threshold that causes no false alarms and the false alarm rate at 99% detection rate for each event. One can see that in contrast to the spoofing attack detection, no event achieves perfect detection (*i.e.*, an EER of 0% or a 100% detection rate at 0% FAR) and only two events achieve detection rates higher than 90% at a false alarm rate of zero. The stark difference to spoofing attack detection can be explained by the flipped class imbalance which leads to masking having a much larger source for false alarms (every 1-second window during our 6-day test period that does not contain an actual event could trigger a false alarm). This also means that even a very low false alarm rate can correspond to a fairly large number of false alarms

per day. Furthermore, masking attack detection is more difficult as the unknown timing of the event means that the window used for feature computation will not be centered on the event time (conversely to spoofing, where the claimed event time is known). Recall, that for event spoofing this uncertainty can only be in the order of milliseconds due to network delays, whereas it can be up to one second for masking detection due to the periodic computation of sensor features.

Nevertheless, for half of the 22 events the majority of masking attacks are detected without causing any false alarms. Some events (*e.g.*, door opened and light off) only exhibit a very low number of false alarms at a detection rate of 99% despite having a very bad detection rate at false alarm rate zero. Therefore, we additionally list the detection rate at a threshold that causes less than one false alarm per day on average. If this limited number of false alarms can be tolerated, the detection rates are improved drastically—with 8 events achieving detection rates higher than 90%.

In order to investigate this more closely, Figure 5.9 shows the tradeoff between DR and the average number of false alarms per day for select events. For some events (*e.g.*, door opened and light off) a singular false alarm affects the performance. If this singular false alarm can be tolerated, a perfect detection rate can be achieved for these events. Other events (*e.g.*, door closed and coffee) have a low number of events with very poor classification scores which lead to a fast rising number of false alarms if the threshold is adjusted to correctly detect them as masked events. A small number of events (*e.g.*, window closed) exhibit such a bad performance that trying to detect even just a single masking attack would lead to many false alarms per day.

5.7.4 Opportunistic Attacker

The goal of the attacker’s surrogate model (see Section 5.6.6) is to accurately predict moments when a spoofed event will be incorrectly verified by the system or when the system will miss the presence of a masked event (we call these moments *opportunities*). Our approach for identifying opportunities is identical for masking and spoofing. As such, in this chapter we focus on the analysis of spoofing attacks.

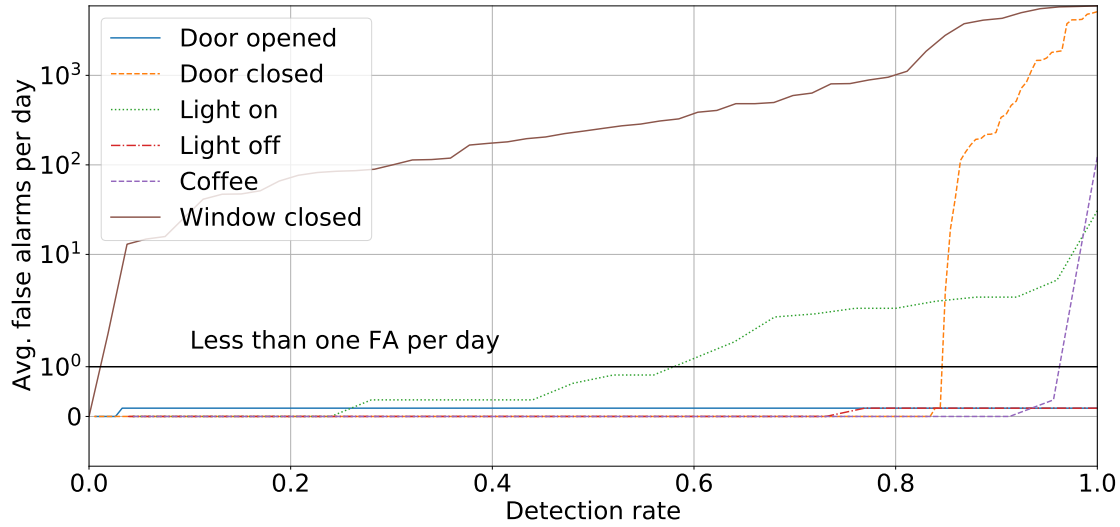


Figure 5.9: Average false alarm counts per day at different detection rates for masking attacks targeting various events. The black horizontal line shows when false alarms reach an average of one false alarm per day over the testing period.

Table 5.8: Error rates for the masking attack detection of different events given in percentages. Numbers in brackets signify the the average number of false alarms per day.

Event	EER	DR(FAR=0)	FAR(DR=99%)	DR(#FAs per day < 1)
Door opened	0.00	2.63	0.00 (0.17)	100.00
Light off	0.00	73.08	0.00 (0.17)	100.00
Screen on	0.00	54.29	0.00 (0.33)	100.00
Radiator on	0.00	0.00	0.00 (3.00)	63.16
Window shade down	0.00	0.00	0.00 (3.33)	65.62
Light on	0.03	24.00	0.04 (30.67)	56.00
Screen off	0.08	0.00	0.05 (43.67)	96.26
Doorbell used	0.14	3.57	0.14 (129.17)	3.57
Coffee machine used	0.15	91.30	0.16 (122.67)	95.65
Radiator off	0.16	4.76	0.16 (139.83)	4.76
Fan on	0.29	99.40	0.00 (0.00)	99.40
Fan off	0.53	66.87	0.00 (0.33)	99.39
Camera on	0.74	76.39	1.36 (1143.33)	94.44
Camera off	0.82	0.00	1.94 (1628.67)	0.00
PC on	1.00	87.18	1.00 (803.50)	87.18
Window shade up	1.39	12.50	1.42 (1037.50)	80.00
Fridge closed	1.99	82.76	4.68 (4077.00)	86.21
PC off	2.01	53.73	2.07 (1675.17)	61.19
Fridge opened	2.20	81.25	5.41 (4595.67)	87.50
Door closed	3.27	83.42	5.78 (4953.67)	84.42
Window opened	4.06	0.00	6.99 (5307.33)	14.71
Window closed	7.20	0.00	8.42 (5969.00)	0.00

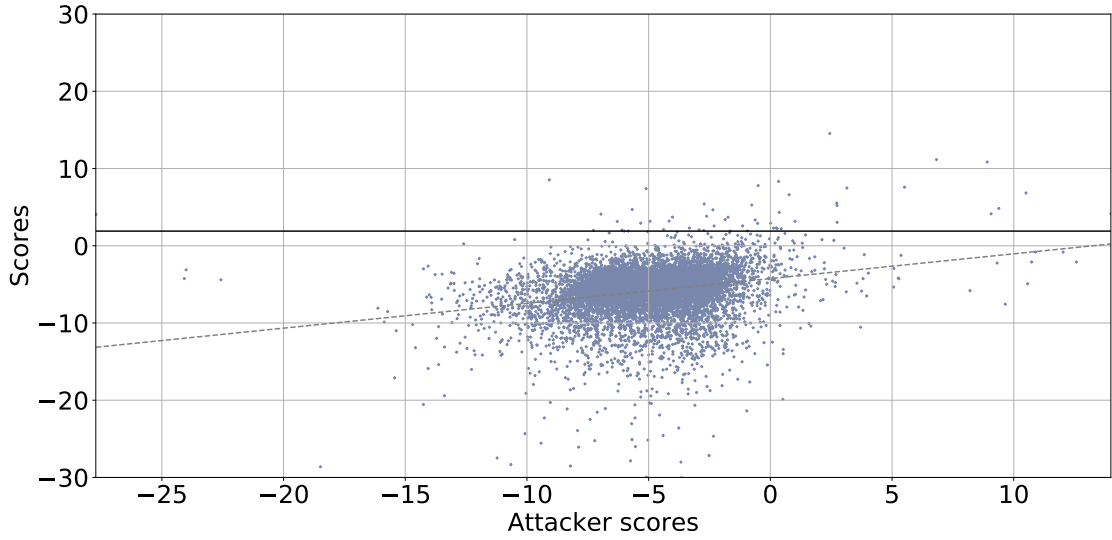


Figure 5.10: Score correlation between the legitimate model and the opportunistic attacker’s surrogate model. The dashed grey line is the linear regression line and the solid black line shows the 99% detection rate cut-off.

Table 5.9: Correlation between the scores produced by the legitimate model and the attacker’s surrogate model.

Event	r-value	p-value
Door closed	0.29	0.00
Fridge closed	0.26	0.00
Window closed	0.21	0.00
Screen off	0.18	0.00
Window opened	0.11	0.00
Doorbell used	0.11	0.00
Door opened	0.10	0.00
Screen on	0.02	0.00
Radiator off	0.01	0.00
Radiator on	0.01	0.00
Window shade up	0.01	0.00
Coffee Machine used	0.00	0.04
Window shade down	0.00	0.09
PC off	0.00	0.25
Fridge opened	0.00	0.78
PC on	0.00	0.98
Camera off	-0.00	0.14
Fan on	-0.00	0.88
Camera on	-0.01	0.00
Fan off	-0.04	0.00
Light off	-0.04	0.00
Light on	-0.11	0.00

We first measure the quality of the surrogate model by measuring the correlation between the scores produced by the surrogate model and those of the legitimate model. If this correlation is high, the attacker is more likely to correctly identify opportunities. A visualization of this correlation for the window open event is shown in Figure 5.10. It is evident that most opportunities (*i.e.*, samples above the 99% DR cut-off) score relatively highly on the attacker’s model. A complete list of correlation coefficients (r-values) is given in Table 5.9. Door, fridge and window and screen events show a particularly high correlation. For these events, we show in Figure 5.11 how successful the attacker can be depending on how conservatively they choose his opportunity threshold. For the lowest threshold (corresponding to the highest-scoring 0.5% of samples), the success rate increases from the baseline 1% to over 50% for the fridge closed event. While 0.5% would suggest an opportunity roughly every 200 samples, this is not true in practice because opportunities do not arise randomly, but typically occur in bursts. Table 5.10 shows an analysis of the median time an attacker has to wait for an opportunity depending on the chosen threshold value. For comparison, the zero-effort attacker achieves an average 1% success rate with no need to wait. Note that both Table 5.10 and Figure 5.11 are computed for a fixed 99% DR to facilitate comparing events with different EERs.

This analysis shows that opportunities for most events are extremely difficult to predict in a timely manner since their physical event signatures almost exclusively follow the event itself. As a result, the attacker could only detect the opportunity after it happened, at which point it will be too late to spoof an event. In addition, the attacker can only be successful if there are any opportunities *at all*. This means that for events with a 100% detection rate, the attacker will always be unsuccessful.

5.7.5 Sensor-compromise Attacker

The sensor-compromise attacker tries to reduce the DR of the system by manipulating sensor readings. A decision making approach that weights some verification sensors more heavily than others is more susceptible to an attacker who can compromise those sensors and change their data. In order to find the worst-case

Table 5.10: Performance of the opportunistic attacker: For each threshold, the table shows the median time until an opportunity is found (tto) and the corresponding success rate (SR) in percentages.

Event	th=0.5%		th=1%		th=5%	
	tto	SR	tto	SR	tto	SR
Fridge closed	470	56.38	313	37.92	130	10.13
Screen off	727	34.20	344	18.82	48	5.50
Window closed	1100	30.67	552	24.00	85	9.92
Door closed	1725	26.70	230	22.91	26	9.42
Fridge opened	2443	16.75	965	10.91	129	3.45
Light off	1639	16.38	998	10.15	111	3.97
Doorbell used	6567	8.92	5823	7.21	2915	4.17
Fan off	912	7.94	364	11.43	49	7.82
Light on	540	7.84	383	5.77	151	2.05
Window opened	413	3.50	178	2.82	43	1.50
Window shade up	440	3.44	193	2.21	29	1.33
Fan on	398	2.19	155	1.55	51	1.11
Door opened	1584	1.85	980	2.08	91	1.77
Coffee Machine used	1694	1.49	1264	1.89	80	1.58
PC off	267	1.38	135	1.36	30	1.29
Window shade down	227	1.16	115	0.95	26	1.06
Camera off	230	1.12	114	1.15	25	1.07
Camera on	245	1.06	126	1.23	24	1.09
Radiator on	1091	1.04	331	1.24	58	1.09
Screen on	400	0.80	175	0.72	33	0.89
PC on	258	0.64	128	0.67	26	0.81
Radiator off	595	0.41	224	0.66	44	0.93

performance of the combined SVM classifier in the presence of a sensor-compromise attacker, we test all possible combinations of a subset of C compromised verification sensors in order to find the subset of compromised sensors that has the biggest impact and reduces DR the most. This means that for each C smaller than the total number of sensors, we choose the combination of sensors that results in the lowest DR.

For the fusion-based approach a theoretical lower bound and a heuristic-based upper bound are given instead. The lower bound is the detection rate of the system when a sensor which always predicts that no attack has occurred is added to the system. The upper bound is computed by choosing a selection of compromised sensors using a heuristic. For a step-wise increasing number of compromised sensors,

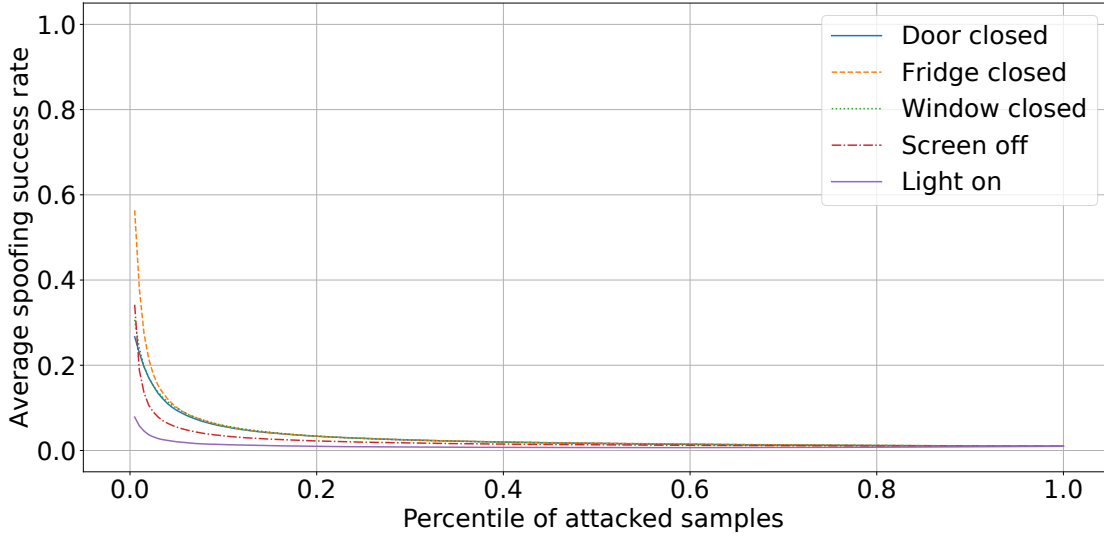


Figure 5.11: Success rates of the opportunistic attacker for various thresholds. A higher thresholds leads to higher attack success rate but increased wait time for an opportunity.

the used heuristic selects the next sensor based on which sensor reduces the detection rate the most. If no sensor reduces the detection rate at the current number of compromised sensors, the sensor with the lowest false alarm rate is chosen. We use a heuristic instead of an exhaustive search as this would be infeasible for larger numbers of compromised sensors. We will later show that this heuristic performs well in practice and leads to fairly tight lower and upper bounds.

The attacker replaces the feature data of the C affected sensors before the test data is passed on to the classifier or the fusion rule according to the method described in Section 5.6.6. While we only present results for spoofed events, the attack works analogously for masked events. It is also possible to combine compromised sensors with waiting for opportunities, although the power of compromised sensors makes this unnecessary for most events. Table 5.11 shows that an attacker that is able to compromise only a few key sensors is able to reduce the DR of the combined SVM to zero without requiring opportunities. We are thus interested in the minimal number C_α at which the attacker is able to reduce the detection rate to zero and thus blind the system.

In Figure 5.12, the worst-case detection rate at false alarm rate zero is shown for both the combined SVM and the single-sensor SVM fusion approach when different

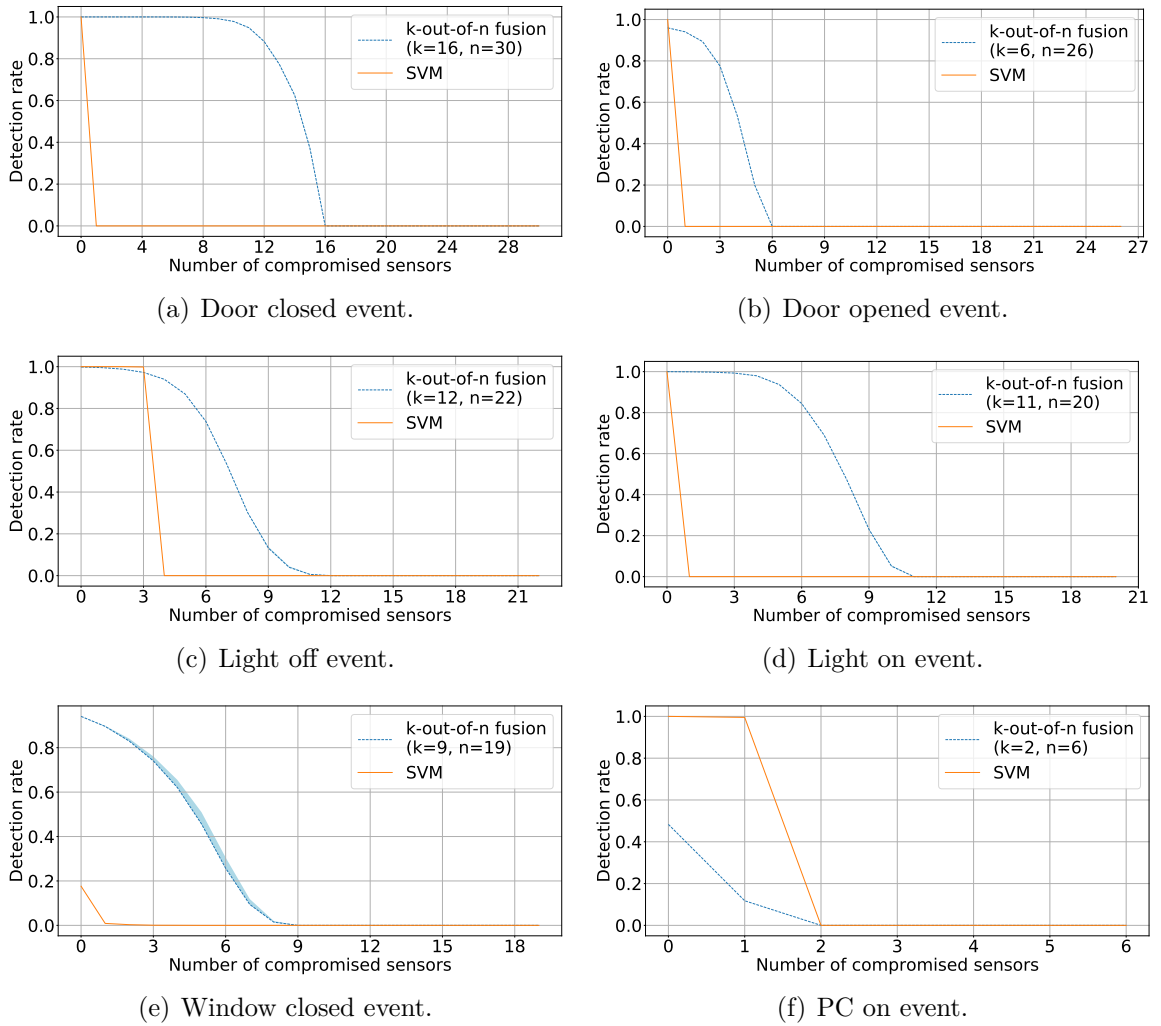


Figure 5.12: System performance against a sensor-compromise attacker with C sensors under their control for various events. Detection rates are given for the combined SVM and the single-sensor fusion approach for different numbers of C and with a false alarm rate of zero. For the fusion-based approach an upper and a lower bound are given to make computation tractable. The lower bound is displayed as a solid line whereas the upper bound is shown as shaded area. More details can be found in the text.

numbers of sensors are compromised. As mentioned earlier, the worst-case detection rate of the fusion scheme is constrained by a theoretical lower bound and a heuristic-based upper bound. It can be seen that these bounds are fairly tight in practice. Many events (*e.g.*, door or light events) have detection rates for the combined SVM and the fusion approach that are very close when no sensor has been compromised. For some of these events (*e.g.*, the door closed event) the detection rate of the combined SVM drops to zero if even just a single sensor has been compromised.

In contrast, the fusion approach is still able to detect some attacks when half of the verification sensors are under the attacker's control (*e.g.*, the attacker needs at least $C_\alpha = 16$ out of the 30 sensors used to detect the door closed event to blind the system). Other events (*e.g.*, the light off event) initially have a higher DR for the combined SVM than the fusion approach (as sensors with lower accuracy are given equal weight to more informative ones), but can still only tolerate a smaller amount of compromised sensors before the system is blinded (*e.g.*, $C_\alpha = 4$ for the combined SVM versus $C_\alpha = 12$ for the fusion approach in the case of the light off event). For a few events (*e.g.*, the window closed event) the fusion approach can even improve upon the combined SVM's performance when not a single sensor is compromised. In these cases, there was likely a systematic change between training and testing data for one sensor that the combined SVM particularly relies on, but which the sensor fusion can compensate for. However, if there are only one or two good sensors aiding in the detection (*e.g.*, for the pc on event), then the sensor fusion approach performs worse than the combined SVM as significant amounts of non-informative sensors are introduced in the decision function.

In Table 5.11, the results are shown for all 22 event sources for both approaches and different values of C . One can see that there is a tradeoff between stronger performance when no sensor is compromised (SVM) and better resilience to sensors falsifying data (sensor fusion). Sensor fusion outperforms a single SVM classifier for 11 out of 22 events with regards to the minimal C_α that the attacker needs to blind the system. Adopting a single SVM classifier only leads to better values for C_α in four cases (doorbell, pc off, radiator on and radiator off), two of which already exhibit very bad performance without an attacker present, and regardless whether a single classifier or sensor fusion is being used. For the other two, the difference in the presence of an attacker is negligible and the main advantage of the single classifier approach is when no attacker is present. Furthermore, events where adopting sensor fusion leads to much worse performance as compared to the combined SVM (*e.g.*, pc and camera events), rely on only one or two strong sensors (*i.e.*, the power meters as they are both corded devices), thus negating the

Table 5.11: System performance against a spoofing attacker that has C compromised sensors under their control. For each event, the table shows the detection rate in percentages at a false alarm rate of zero for both the combined SVM and the single-sensor fusion approach. For the sake of clarity, only the worst-case lower bound is given for the fusion scheme.

Event	C=0		C=1		C=2		C=5		C=10		C=15	
	SVM	Fusion	SVM	Fusion	SVM	Fusion	SVM	Fusion	SVM	Fusion	SVM	Fusion
Coffee machine used	100.00	89.81	0.00	53.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Fan on	100.00	6.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Light off	100.00	99.85	100.00	99.54	100.00	98.81	0.00	86.91	0.00	3.98	0.00	0.00
Fridge opened	100.00	92.81	0.00	86.64	0.00	77.78	0.00	24.48	0.00	0.00	0.00	0.00
PC on	100.00	48.32	99.52	11.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Door closed	100.00	100.00	0.00	100.00	0.00	100.00	0.00	99.99	0.00	97.84	0.00	37.05
Door opened	100.00	95.98	0.00	94.07	0.00	89.38	0.00	19.88	0.00	0.00	0.00	0.00
Screen on	100.00	37.40	0.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Fridge closed	100.00	99.64	0.00	99.20	0.00	98.22	0.00	86.12	0.00	22.61	0.00	0.00
Camera on	99.99	47.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Light on	99.98	99.98	0.00	99.91	0.00	99.74	0.00	93.69	0.00	5.11	0.00	0.00
Camera off	99.90	15.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Fan off	99.82	97.38	0.23	93.57	0.00	48.22	0.00	0.00	0.00	0.00	0.00	0.00
Screen off	99.71	56.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PC off	99.61	13.68	0.03	2.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Radiator on	98.93	36.78	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Window closed	17.63	94.09	0.88	89.59	0.31	82.96	0.02	45.64	0.00	0.00	0.00	0.00
Radiator off	7.65	15.98	1.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Doorbell used	0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Window opened	0.35	81.18	0.00	69.94	0.00	54.88	0.00	8.24	0.00	0.00	0.00	0.00
Window shade up	0.28	77.46	0.06	46.62	0.00	16.96	0.00	0.00	0.00	0.00	0.00	0.00
Window shade down	0.19	14.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

main benefit of using sensor fusion. On the other hand, if there are more sensors contributing to the detection, sensor fusion can even noticeably improve on the performance of a single classifier (*e.g.*, for window and window shade events), if the individual sensors' errors are uncorrelated.

5.8 Discussion

One advantage of using PEEVES for event verification is that it enables close to real-time detection of spoofed and masked events for most types of events. Most events are detected in less than five seconds, as the biggest contributor to detection time is the length of the event's sensor signatures. However, some event sources—such as the coffee machine, the window shade and the radiator—only affect their environment over a longer period of time. This can lead to higher detection times of up to one minute in the case of the radiator.

In this chapter, we have investigated a smart environment located in a single room. Future real-world deployments might cover entire houses and have even

higher numbers of event sources and sensors per room than the setup used in our experiment. We expect our system to scale well when it comes to real-time detection in these scenarios. The biggest issue with regards to scalability stems from the complexity of training the system, especially when a monolithic classifier approach is being used. One solution to this problem could be to limit the verification sensors considered for an event to sensors located in the same room or an adjacent room. We leave this investigation of large-scale deployments for future work.

A related issue is the amount of time necessary to train the system. One way to solve this problem is to create generic sensor signatures based on data collected in other environments. We expect that many of the sensor signatures learned by PEEVES could be generalized and used to bootstrap the training phase in different locations. For example, an opening door has a specific physical effect on air pressure sensors; this effect should look very similar regardless of sensor manufacturer and where the sensor is located in the room. Because of that, we anticipate that the system could start with a general sensor pattern for each event/sensor combination, check if a relevant sensor can be found, and then simply tune the signature to the new deployment by adjusting parameters such as amplitude or duration. This way, PEEVES could drastically cut down the required system training period.

Notably, PEEVES is designed to make use of sensors that are already deployed. In practice, many smart home devices do not yet expose their sensor data with a sample rate high enough to capture the short sensor signatures produced by some events. While we have seen an indication that even low sample rates can be used to create distinguishing signatures—the power meters used in the experiment were unmodified COTS hardware with a sample rate of 1Hz—a firmware upgrade that allows access to higher sample rates might be necessary to achieve more consistently reliable sensor signatures. However, a higher reporting frequency might not always be feasible for battery-powered devices, and for these cases more research into an energy-efficient way of handling this increased amount of sensor data is required.

The system introduced in this work requires devices to report events in a timely manner. If events would be accepted although the time window used for verification

has passed, an attacker with read access to the sensors would be able to determine with near certainty if they could successfully launch an attack. Furthermore, recent work by Goksel *et al.* [209] has shown that even just reordering real occurring events can have security implications. Requiring events to be reported within the time span of usual network delays limits the attackers ability to exploit these effects.

If new devices get added to a smart environment that uses PEEVES or if an existing device gets moved, then the system has to adapt to these changes in the environment. When the monolithic classifier approach is used, this inevitably means that the classifier of each affected event has to be retrained. In the case of an ensemble of sensor classifiers only the affected sensors have to be retrained. In either case, it is important that the capabilities of the attacker are factored into the training process, as the assumption that all devices are uncompromised does not hold when the system is being retrained. We leave the consideration of retraining the system under possible poisoning attacks for future work.

5.9 Conclusion

In this chapter we have presented PEEVES, a system to automatically verify smart home events based on their physical event signatures. We verify 22 events using 48 verification sensors and validate our system against both spoofing and masking attacks on a real-world dataset collected over two weeks.

We show that most spoofing attacks are detected by a multitude of sensors, which highlights that PEEVES can be used even with few off-the-shelf sensors already integrated in smart home devices, although a software update might be required to expose the necessary sensing fidelity. 9 out of 22 events achieve a near-perfect EER of 0.00% and 15 events achieve a 0% false alarm rate at a detection rate exceeding 99.9%.

We further show that masking attacks are more difficult to detect than spoofing attacks due to the increased attack surface. However, for 11 out of 22 events the majority of masking attacks can be detected without causing any false alarms.

Furthermore, 8 events achieve a detection rate of higher than 90% if less than one false alarm per day on average can be tolerated.

We also formulate the notion of an opportunistic attacker that attempts to predict opportunities (*i.e.*, times when a spoofed or masked event will go undetected) based on verification sensor data. This attacker builds a surrogate event verification model based only on data preceding each sample. We show that some events exhibit signatures that precede or overlap with the event, which makes it possible for the attacker to predict opportunities *before* they occur. We note that spoofing opportunities for the fridge, window and door events are easiest to find, although their high detection rate still makes an attack very difficult to execute.

Finally, we investigate the impact of a more powerful type of attacker, relaxing the assumption that verification sensors are secure and thus allowing a subset of sensors to be compromised. We show that the use of a single classifier makes the system vulnerable if even just one key sensor has been compromised—which effectively reduces the attack detection rate to zero. We thus demonstrate how an approach based on the fusion of an ensemble of classifiers trained on the data of single sensors can be used to increase the system’s resiliency against compromised sensors, provided that the detection does not rely on a single sensor.

However, for some events the fusion approach exhibits worse detection performance than the single classifier does. Much of this is due to the fact that a few sensors struggle to reliably detect events in specific contexts, but perform well the rest of the time. In the following chapter, we show how contextual information about smart home device states can be used to better select sensors for verification and to reduce the vulnerability of the system to context-dependent performance decreases.

We make our entire dataset available online to allow researchers to build on our results.

6

Context-aware Sensor Selection in Smart Environments

Contents

6.1	Introduction	142
6.2	Context-aware System Model Extension	144
6.3	Threat Model	146
6.4	Methods	147
6.4.1	Pre-processing	147
6.4.2	Sensor Fusion	147
6.4.3	Context-aware Attacker Behavior	151
6.5	Evaluation	154
6.5.1	Impact of Context-awareness	154
6.5.2	Sensor-compromise Attacker	158
6.5.3	Context Selection and Context Manipulation Attacks	160
6.6	Limitations	163
6.7	Conclusion	164

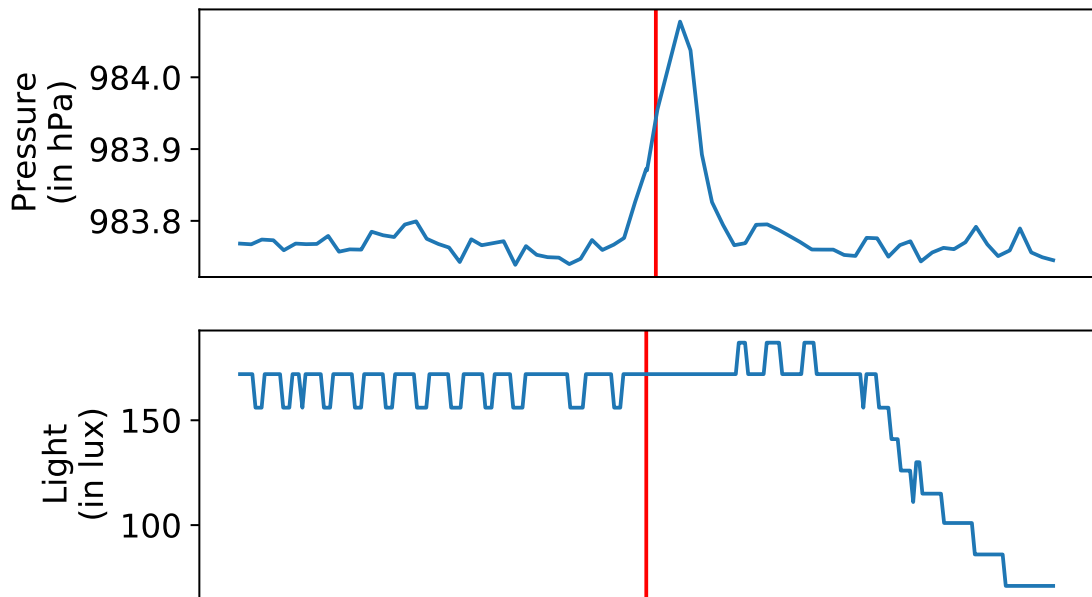
6.1 Introduction

In the previous chapter, we have introduced how a sensor fusion scheme can be used to improve the robustness of a security system that is based on ubiquitous sensing, such as the event verification system PEEVES introduced in Chapter 5.

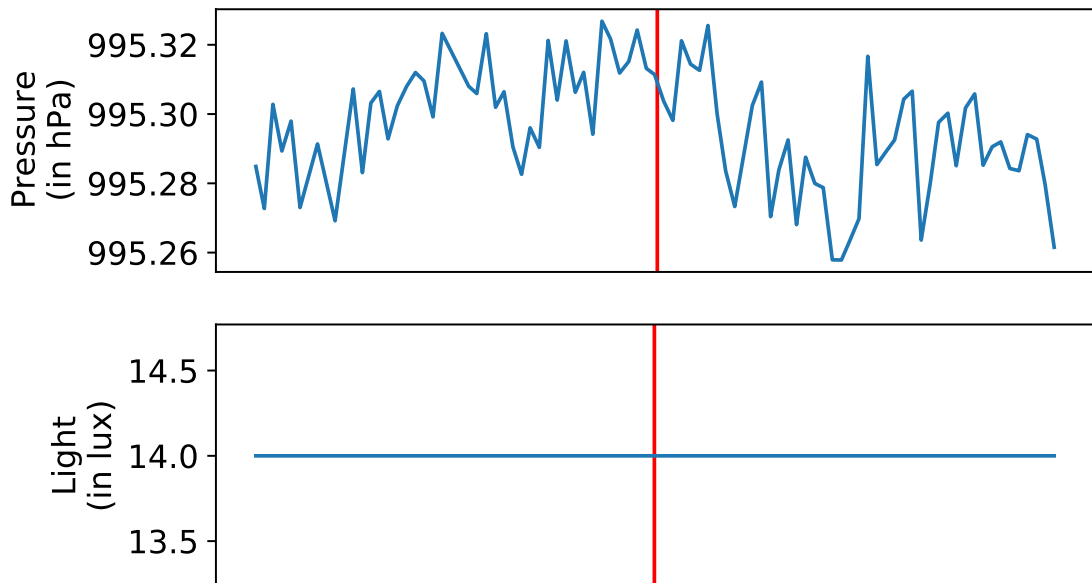
While the fusion scheme proposed in the previous chapter proved to increase the robustness of PEEVES to compromised sensors, the fusion scheme itself can be vulnerable to context-dependent detection performance issues. This is due to the fact that sensors, which are generally useful for the detection of a particular event, can exhibit degraded detection performance if there are changes in the environment. These environmental changes can affect the physical phenomena that the sensor readings are supposed to capture. See Figure 6.1 for a visualization of this phenomenon. For example, a closing door might have an entirely different event signature depending on whether a window in the same room is open or closed. If these differences are not learned or otherwise taken into account, the detection performance of the system will suffer. An open window might lead to a more pronounced “door closed” signature for an accelerometer, whereas an air pressure sensor might not show a perceptible signature with an open window—while otherwise exhibiting good detection performance.

Luckily, in modern event-driven smart homes many of these possible contextual changes are monitored by dedicated sensors and are reported to the system via event notifications. Therefore, the knowledge of these contextual states can be used to improve the detection performance of event verification systems. In this chapter, we use this information to dynamically select sensors based on their past performance in a given context, thus addressing **RQ5**.

However, because it is possible that the event sources corresponding to these contextual states are themselves compromised, it is important to keep in mind how this sensor selection could be misled by an attacker (**RQ4**), as the attacker might be able to spoof or mask events related to the contextual information that is used to decide if a sensor should be selected or not. This could lead the event verification system to assume that the system is in a state that deviates from the



(a) Window closed/Light on



(b) Window opened/Light off

Figure 6.1: This figure gives examples of how different environmental contexts can influence how an event is perceived by a sensor. The red lines correspond to the timing of a door opening event. One can see that there are noticeable effects on both the air pressure sensor when the window is closed and the light sensor behind the door when the light has been turned on. However, with an open window and after the light has been turned off no discernible signature can be perceived for either sensor.

actual circumstances. In order to limit the potential impact of this increase in attack surface, these contextual events should in turn be verified as well—thus creating a *recursive event verification* chain.

In this chapter, we investigate the advantages of enhancing event verification systems with context-awareness, and discuss the involved risks. We introduce a sensor selection scheme that selects sensors based on their prior performance in a given context and that aims to ensure that only appropriate sensors are chosen for event verification. Furthermore, we analyze the role that recursive verification of contextual events plays in ensuring that an increased detection performance does not come at the cost of security when faced with a determined attacker. We base the evaluation of the proposed context-aware extension on the dataset collected within Chapter 5. In summary, we make the following contributions:

- We propose a context-aware extension for event verification in event-driven smart home systems.
- We show that context-based sensor selection can improve the detection performance of these systems (**RQ5**).
- We demonstrate the importance of recursively verifying each contextual event in order to limit an increase of the attack surface (**RQ4**).

6.2 Context-aware System Model Extension

In this chapter, we extend the system model for the event verification system from Chapter 5 and enable this system to leverage information about the current states of other devices to select appropriate verification sensors based on past performances in similar contexts.

Following the example from Figure 6.1(b), such an extended system would stop using the air pressure sensor to detect “door closed” events while the window is open, whereas it would add the light sensor to the pool of verification sensors for the detection of these events when the light is on.

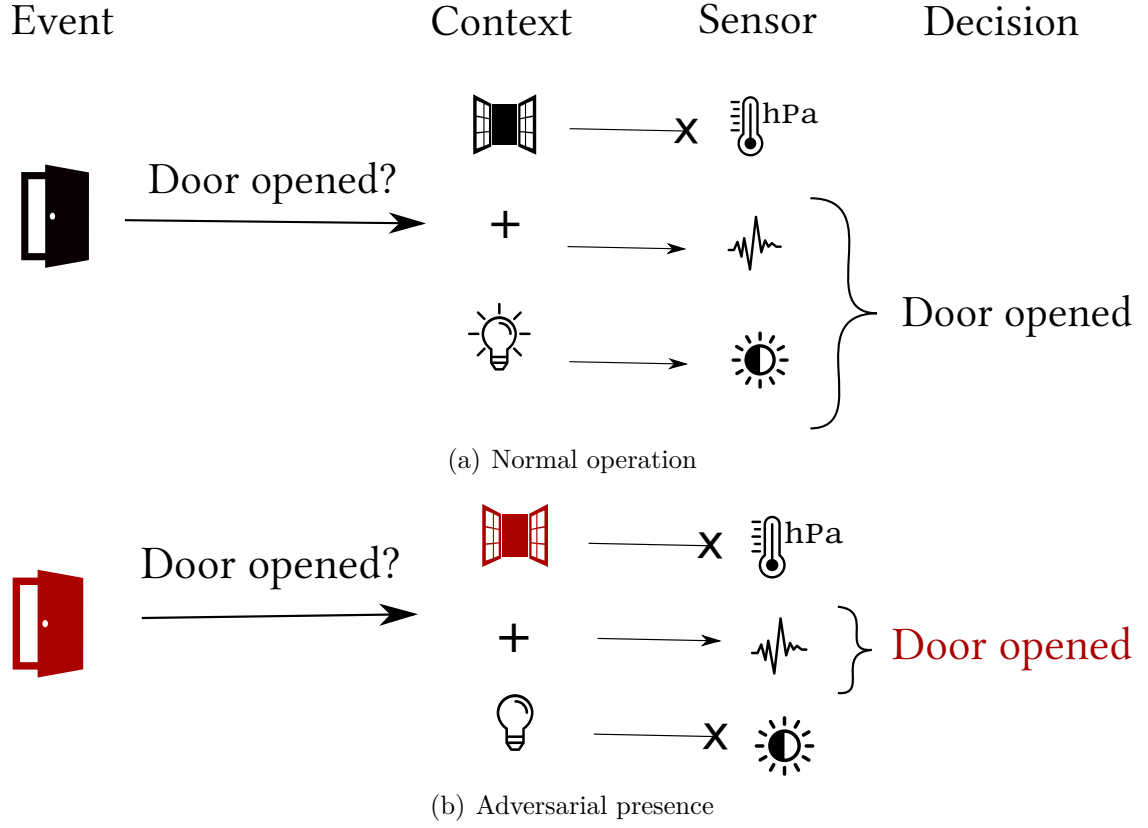


Figure 6.2: Sample system behavior for a “door opened” event during normal operation and in the presence of an adversary. During normal operation, an opened window leads to a deselection of the air pressure sensor whereas a light sensor is added to the selection due to the light being on. An adversary can manipulate the window state (*context manipulation*) to force the deselection of the air pressure sensor, or wait until the light has been turned off (*context selection*) if the deselection of the light sensor would be beneficial for them. In either case, the accelerometer is unaffected by the current context.

To this end, each event notification type is associated with a corresponding set of relevant context states $\mathcal{C}$ and there exists a function $f : \mathcal{C} \rightarrow \mathcal{V}$ that maps the current context to the verification sensors $\mathcal{V}$ used for the event verification.

We assume that event sources and verification sensors remain uncompromised during training, and that the system hub remains uncompromised both at training time and during operation.

An overview of the extended event verification system is given in Figure 6.2(a).

6.3 Threat Model

As in the preceding chapter, the attacker can remotely compromise event sources in the smart home. Therefore, the attacker has the ability to choose if and when they send out an event notification to the hub. Hence, the attacker can spoof or mask events of the compromised event source during operation of the system. The attacker is still bound by the system requirement of timely event notification and can not delay their decision to launch an attack. They are also unable to make physical changes to the compromised devices or events.

In order to evaluate the increased attack surface for context-aware systems, we extend the threat model from Chapter 5 to include two new types of attackers in addition to the zero-effort attacker:

- *Context-selective attacker*
- *Context-manipulating attacker*

Besides the capabilities described above, these new types of attackers have knowledge of which event states are used as context information by the system and which sensors are selected by the system in a given context. They also know which context state the system is currently in, and are able to read the data of the verification sensors. This enables the context-selective attacker to launch attacks in a context of their choice (when they expect an attack to be more successful). If the attacker, for example, knows that the system is less likely to detect the occurrence of a “door opened” event when the light is off—because the light sensor behind the door is less reliable in that context and will not be used by the system—then the attacker might decide to hold off, if possible, on launching a masking attack until the light has been turned off. The context-manipulating attacker, on the other hand, can do more than just time the attack, as they can also spoof or mask context events to force the system into a desirable state that makes subsequent spoofing or masking attacks more likely to succeed. For example, consider an attacker who wants to spoof a “door opened” event. If the attacker knows that

this event is more vulnerable when the system erroneously accepts that the window is open—because this leads it to exclude a descriptive air pressure sensor from the selection of verification sensors—then before moving on to the actual target of the attack, the attacker might first spoof a “window open” event, as this would reduce the number of reliable sensors used for verification. See Figure 6.2(b) for a visualization of how these two types of attackers would affect the system.

6.4 Methods

6.4.1 Pre-processing

We use the same pre-processing steps as for the original PEEVES system in Chapter 5 for data labeling, dataset division, sensor window selection, and feature extraction. In the following, we present a brief recap of the PEEVES pre-processing steps.

Separately, for each event, all points in time are labeled with 1 (event occurred) or 0 (no event). The 13-day dataset is then sequentially split into three parts: development, training and testing. The one day long development set is used to calculate sensor-specific windows and to select appropriate features. The remaining twelve days are evenly split for training and testing purposes. Apart from providing data to train the classifiers, the training set is additionally used to compute context-dependent classifier performance (*cf.*, Section 6.4.2).

After dividing the dataset, we then follow the sensor window selection and feature extraction processes described in Section 5.6.

6.4.2 Sensor Fusion

In this chapter, we use a sensor-fusion based approach similar to the one introduced in Chapter 5, as this enables us to dynamically choose which sensors are used for the decision. Related work has investigated how context-based classifiers [210] can be used to adapt a classification system to dynamic changes in the environment. However, using a monolithic SVM classifier would make it impossible to do this without retraining, as this type of classifier requires constant-length feature vectors, *i.e.*, we can not test with fewer features (or sensors) than the classifier has been

trained with. In fact, the approach proposed in [210] requires separate classifiers for each considered context. Furthermore, a single classifier is prone to misclassification when a single sensor, which showed good performance during training, deteriorates later on during operation. A sensor-fusion based approach is more likely to be robust to this kind of problem if there is a sufficient number of adequate sensors.

The proposed sensor fusion approach trains classifiers on the data of each verification sensor, separately. The decisions of these classifiers are then combined using a fusion rule. This allows us to select or deselect sensors based on the current context and the states of other devices. In this chapter, we only consider contexts that correspond to a single stateful event source. A stateful event source is, *e.g.*, the door sensor with corresponding context states “door is open” and “door is closed”. Most event sources in the PEEVES dataset are stateful event sources with the exception of the doorbell and the coffee machine as these are stateless sporadic events which can not be used for context information.

Single-sensor Classifiers

For each verification sensor for which we have extracted features, we train one classifier. These classifiers are trained on all the features derived from this sensor’s data. Similar to PEEVES we choose binary linear support vector machines (SVMs) parametrized by the penalty parameter C for these single-sensor classifiers. We assign different values of C proportional to the fraction of samples for each class in order to deal with the class imbalance of events and non-events. The decisions made by each of these single-sensor classifiers are then passed on to the sensor fusion procedure.

Context-aware Sensor Selection

An automated event verification system can only be useable if it runs transparently in the background and if false alarms are kept to a minimum. We therefore use an approach based on the conditional false alarm rate to select sensors for the context-dependent fusion. This choice is motivated by the observation that the FAR can be much higher inside of problematic contexts than outside of them. See Figure 6.3 for an example of this effect. The conditional false alarm probability

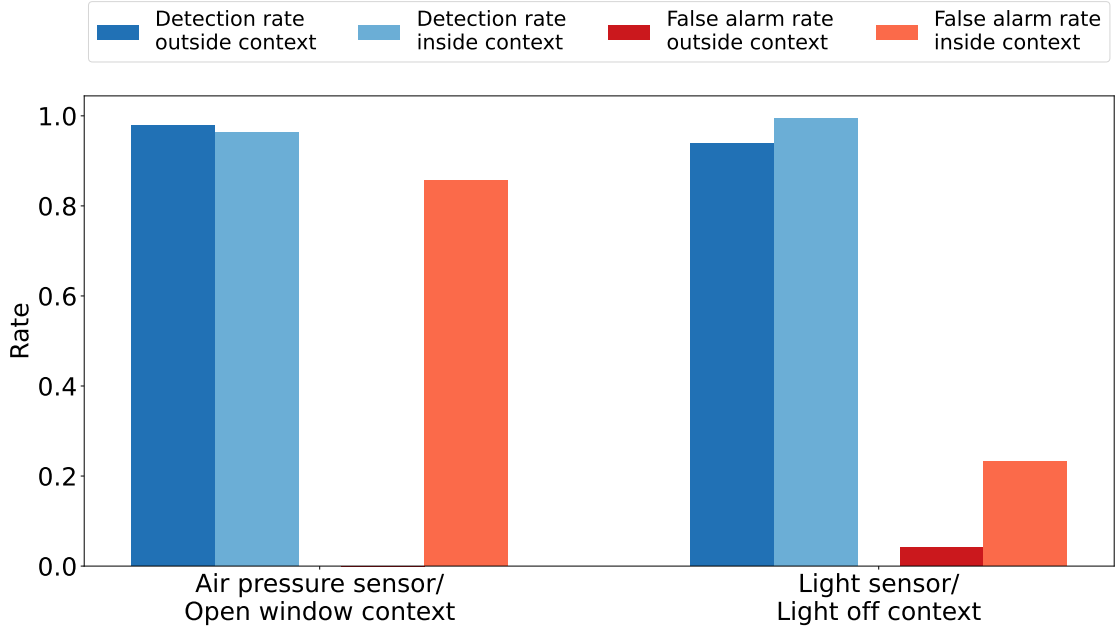


Figure 6.3: Illustration of context-dependent drops in performance. Displayed are the detection rates and false alarm rates for the “door opened” event for two different sensors (an air pressure and a light sensor) both inside and outside of a context (“window open” and “light off” context, respectively) that impacts their false alarm rates. Both of the sensors exhibit a higher false alarm rate within their respective problematic contexts than outside of them.

(PCFA) is defined as the probability that the system erroneously reports an attack if a specific context is given.

We define the PCFA as follows for a classifier S and feature vector F_t at time t :

$$PCFA := Pr(S(F_t) == 1 | y(t) == 0, \mathcal{C})$$

where $y(t)$ is the ground truth at time t and $\mathcal{C}$ is contextual information.

The selection metric is first computed for each single-sensor classifier without considering any context in order to establish a baseline performance of the classifier. Then, we compute the selection metric for each available context (*i.e.*, a stateful event source). In the following, it is assumed that a lower value is better with regards to the selection metric.

For each event, we choose a threshold value on the classification performance

that each sensor classifier has to achieve to be considered for the fusion. The default selection of sensor classifiers is based on the baseline performance. Then, based on the current context, sensor classifiers that are typically selected by default, but which are performing worse than the threshold are deselected, whereas sensor classifiers which are not part of the default selection, but have better performance than the threshold, are added to the pool of classifiers. Additionally, a minimal number of context occurrences can be chosen to make sure that the context-aware selection is not based on too few data points. If there were too few context occurrences in the training set for the current context, the baseline performance is used exclusively to make the selection.

This process gives us a function $f : \mathcal{C} \rightarrow \mathcal{V}$ which maps the current context $\mathcal{C}$ to the set of verification sensors $\mathcal{V}$ that the system should use for verification.

Fusion Rule and Metrics

Similar to the fusion scheme in Chapter 5, we use a k-out-of-n fusion rule which decides that an event has occurred (is labeled as 1) when at least k out of the n sensors used for the fusion agree that it has occurred [207]. The fusion rule is defined as follows:

$$\sum_{i=0}^{|\mathcal{V}|} S_i > k$$

where $\mathcal{V} = f(\mathcal{C}) = \{S_0, S_1, \dots, S_n\}$ is the set of verification sensors used in the current context $\mathcal{C}$.

As before, we choose this fusion rule on account of its recognized strong performance and ease of use [208]. Additionally, the parameter k allows us to control the tradeoff between detection rate (DR) and false alarm rate (FAR).

Similar to the previous chapter, for spoofing attacks the attacker flips a 0-sample (non-event) to a 1-sample (event). The DR is therefore defined as the fraction of 0-samples that are correctly classified (*i.e.*, the probability of detecting a spoofing attack conducted at a random time). Conversely, the false alarm rate is defined as the fraction of wrongly classified 1-samples (*i.e.*, actual events that are wrongly

classified as spoofed). For masking attacks, the attacker attempts to do the opposite (change a 1-sample to a 0-sample). The DR is therefore defined as the fraction of correctly classified event periods (*i.e.*, the probability of detecting a masking attack of an actual event in a time period equal to the safety margin around the event), whereas the false alarm rate is defined as the fraction of wrongly classified 0-samples (*i.e.*, the probability of wrongly classifying the occurrence of a masked event at a random point in time).

6.4.3 Context-aware Attacker Behavior

To model the potential behavior of a context-aware attacker, we employ the use of attack trees [211]. Attack trees use nodes to model intermediate steps or sub-attacks that the attacker launches in order to achieve their ultimate goal, situated at the root node. The nodes of attack graphs can have assigned values corresponding, for example, to the probability that the attacker can successfully execute a sub-attack [212] associated with this node. Through this augmentation with attack probabilities it is possible to compute the probability of success for the attacker when they follow a certain *attack path* and execute the corresponding sequence of attacks.

In our case, the root node of an attack tree is the event that the attacker intends to spoof or mask. Leaf nodes are context states that the system can be in before the attack has started. For example, if the attacker wants to spoof a “door opened” event, possible context state leaf nodes could be “window is open”, “light is on”, or the “default” context for all possible system configurations.

In our analysis, we limit ourselves to a single context at a time and only consider leaf nodes consisting of a single context state. While it is possible that combinations of context states can affect the detection performance, considering all combinations of context states would drastically increase the complexity of the analysis. Furthermore, occurrences of specific combinations of context states can be very rare and are thus at risk of not being properly represented in the dataset. However, our analysis can be extended to account for specific context combinations should the need arise and should sufficient data be available.

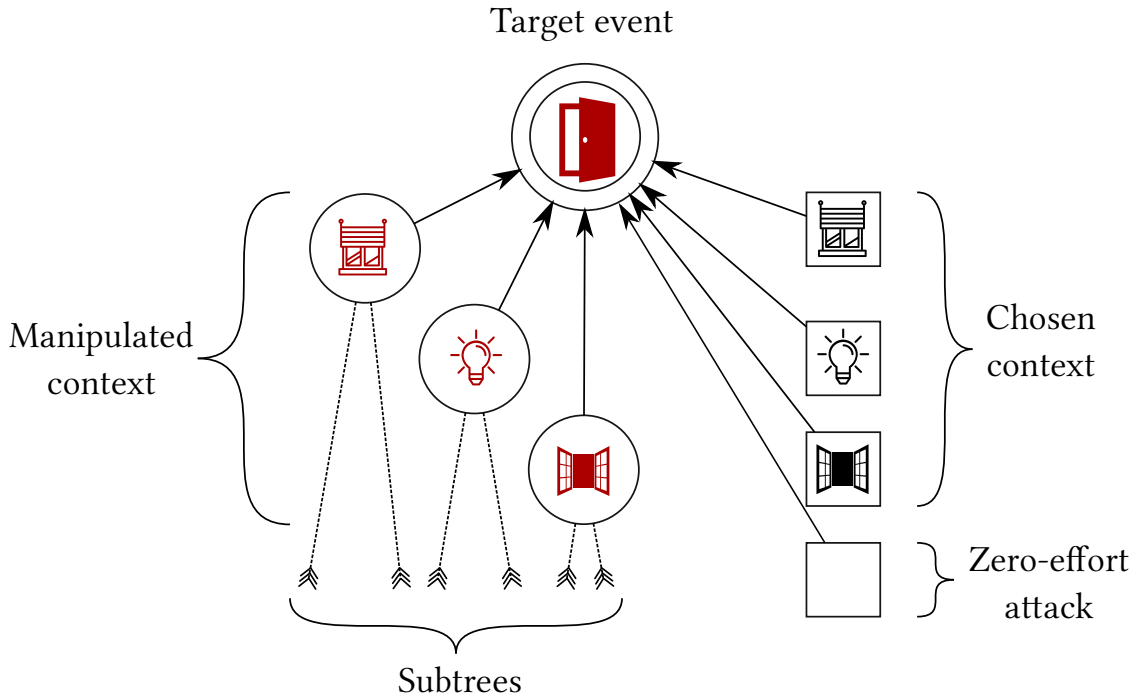


Figure 6.4: Sample attack tree for a “door opened” spoofing attack. The target event is situated in the root node. Directly connected leaf nodes represent specific contexts, which can be exploited by the context-selective attacker, and a default context that represents the behavior of the zero-effort attacker. Each sub-tree represents a context which has been manipulated by a spoofing or masking attack, and can be unrolled recursively as a separate attack target.

A single instance of a spoofed or masked event is represented by an edge going from the originating system state to the state targeted by the attack. Edges connecting leaf nodes directly to the root node represent attacks that do not make use of any spoofed or masked context. Intermediate nodes constitute sub-attacks where the attacker spoofs or masks a context to force the system in a state beneficial for subsequent attacks. These intermediate nodes are the root nodes for recursively defined sub-trees with their own leaf nodes and intermediate nodes corresponding to the context event source in question. If we recall the three attacker types from Section 6.3, the zero-effort attacker can be understood as only starting attacks in the default context leaf node, whereas the context-selective attacker can pick one of the other context-specific leaf nodes to start their attack based on their current needs. Finally, the context-manipulating attacker can make use of the leaf nodes of any of the sub-trees, and spoof or mask context events to reduce the

likelihood of getting detected by the system. Please refer to Figure 6.4 for a graphic representation of intermediate states and sub-attack trees.

If more than one recursion level is considered and if contexts are reused in different sub-trees, sequence order has to be taken into account when computing the success probabilities for attack paths. This did not turn out to be an issue in our case, as all attacks with more than one recursion level had non-overlapping contexts. As a further simplification, we therefore assume that consecutive steps are time-independent and each sub-attack only depends on the current state of the system. Nodes are augmented with the associated probability of success that the attacker can expect when executing a sub-attack originating from this system state.

We recursively define the attack tree A_e for a spoofed event e with a set of context states $\mathcal{C}$ as follows:

$$A_e := (e_s, \mathcal{A}, \mathcal{L}, \mathcal{E})$$

where e_s is the root node of the attack tree representing the target event, $\mathcal{A} = \mathcal{C}_S \cup \mathcal{C}_M$ is the set of sub-attack trees corresponding to both spoofing and masking attacks on context states, $\mathcal{L} = \mathcal{C} \cup \{\epsilon\}$ is the set of leaf nodes representing context states for the context-selective attacker as well as the zero-effort attacker (the default state ϵ), and $\mathcal{E}$ is the set of edges connecting each sub-attack tree and each leaf node to the root node e_s . The attack tree for a masked event is defined analogously.

To compute the best-case attack path for the attack as a worst-case detection performance metric for the event verification system, we backtrack starting from the root and recursively compute the performance for sub-attacks.

First, we compute the highest success probability that the attacker can achieve through edges directly connecting a leaf node with the root node. Then, for each sub-attack tree, we compare the success probability under the assumption that the sub-attack was successful. If this success probability is higher than the success probability for the zero-effort attacker starting from the default context leaf node, then we recursively determine the success probability for the sub-attack and compute the resulting attack success probability using Formula 6.1. Otherwise, we

skip this sub-attack tree to limit unnecessary computations and comparisons as the corresponding attack path can not have a higher success rate than the zero-effort attacker. Therefore, the attacker would choose to spoof or mask the event without first attacking one of the context states.

Once we have completed this process for all sub-attack trees, we obtain the attack path with the maximum attack success probability as the best-case attack success rate (SR) for the attacker.

$$SR = \max_{\forall p=(v_1, \dots, e_s)} \cup_{v_i \in p} Pr(v_i) \quad (6.1)$$

where p is a path from a leaf node (potentially of a sub-tree) to the root node and $Pr(v)$ is the probability that the sub-attack originating in the system state corresponding to node v is successful. Conversely, the worst-case detection rate can be obtained with $DR = 1 - SR$.

This analysis can be further extended to account for additional types of attackers, such as the opportunistic attacker from Chapter 5, using parameterized attack trees [213]. In parameterized attack trees edges are marked with attacker capabilities. This enables an analysis in which edges corresponding to a specific attacker capability can be marked as active or inactive depending on the chosen attacker profile. However, this extended evaluation is beyond the scope of this work.

6.5 Evaluation

6.5.1 Impact of Context-awareness

In order to study to what extent the context-aware sensor fusion extension can improve event verification systems, we compare its performance against the PEEVES sensor fusion scheme from the previous chapter. Figure 6.5 shows the detection rates at a false alarm rate of zero with an optimal set of system parameters for PEEVES with sensor fusion, as well as the context-aware extension. Due to the limitations of our dataset, these parameters have been chosen on the testing set for the purpose of this evaluation. Although this goes against standard practice, as it

can lead to overfitting, it was necessary due to the limited occurrences of relevant contexts within the dataset. Furthermore, it allowed us to study the possible performance gains in an idealized setting. In practice, a larger developmental and training dataset would be required to avoid these issues. This could, for example, be implemented through periodic reevaluation and retraining of the system. However, this is beyond the scope of this work.

To separate the effect of the more stringent sensor selection from the context-aware approach, the detection rate is also given for the case when only the baseline sensor selection is used for the sensor fusion and no context is considered during classification. The detection rates are very high for most events with 9 events having detection rates of higher than 90% for all three system variants. One can see that 19 out of the 22 events already exhibit a large improvement through the improved sensor selection. Additionally, it can be seen that adopting a context-aware approach can further improve the detection rate in seven cases.

In some cases, this is because a sensor that shows generally bad performance is added back into the fusion in specific contexts. Such a sensor is sometimes needed when the main sensors become unreliable, as is the case for the “pc off” event. In the case of the context-free baseline this means that this sensor has to be used throughout the entire operation of the system—thus reducing the overall detection performance—in order to avoid false alarms. For the context-aware approach, on the other hand, the use of this sensor can be limited to contexts where it is necessary, leading to a performance boost of 61%. But this should be done with great care, as it can also lead to reduced performance, as it does in the case of the “fan on” event where a badly performing sensor which is not used by the context-free baseline is added back in by the context-aware approach, leading to a drop in performance of 3%. However, the biggest apparent boost seems to stem from the more stringent sensor selection.

Nevertheless, an attacker who knows that the system is more vulnerable in specific system states is able to exploit this fact to their advantage if they can

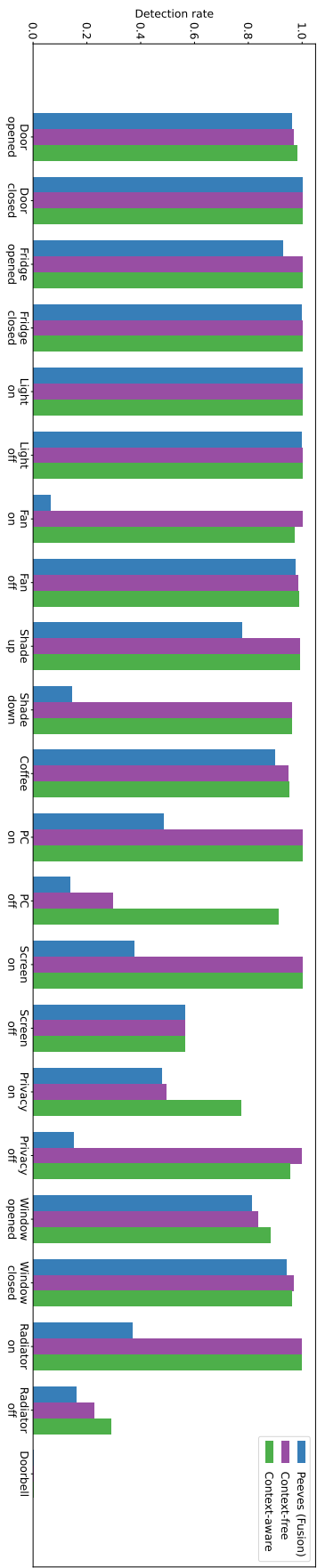


Figure 6.5: Comparison of detection performances for events with regards to event spoofing. Bars correspond to three different event verification system variations: the PEEVES fusion scheme from Chapter 5, an extension that does more stringent sensor selection as a baseline for the new system but remains context-free, and the context-aware approach proposed in this chapter. For all three system variants, the detection rates at false alarm rate zero are given.

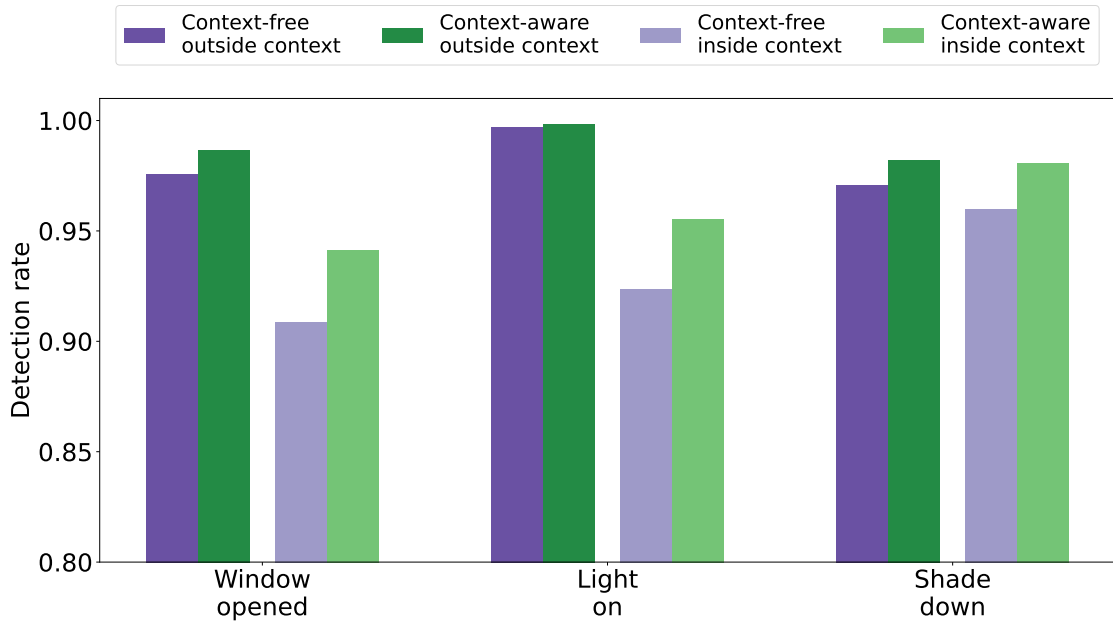


Figure 6.6: Comparison of the performance gain that can be achieved inside a specific context when that context is taken into account in the decision making process. Displayed are the detection rates at false alarm rate zero for the “door opened” event in the top three contexts with regards to event spoofing. The bars correspond to the performance of the context-free baseline both inside and outside the context, as well as the increased performance inside a context when the context-aware system extension is being used.

determine the current context. Figure 6.6 compares the performance of the context-free enhanced sensor selection baseline against the context-aware extension for the “door opened” event in the three most important contexts for this event. This figure shows the detection rates at a false alarm rate of zero both in and out of relevant contexts to demonstrate the improvement that can be achieved through a context-aware approach, as well as the context-dependent vulnerability of the context-free baseline. One can see that an attacker who can choose the context in which they attack, can reduce the risk of getting detected by 1.1–7.3% in the three main contexts for the context-free approach, but this benefit is reduced to 0.1–4.5% when the context-aware extension is used as there is a larger detection rate increase within those contexts.

6.5.2 Sensor-compromise Attacker

The benefit of using the context-aware extension becomes even more clear when we also consider the possibility of sensors being compromised as in the previous chapter. Figure 6.7 shows the worst-case detection rates at false alarm rate zero against a sensor-compromise attacker for the original PEEVES system with a monolithic SVM classifier, the fusion approach from the previous chapter, the extended context-free sensor selection, as well as the context-aware extension illustrated with six different events. Similar to Chapter 5, we give a theoretical lower bound and a heuristic-based upper bound for the fusion-based approaches. The lower bound is the detection rate of the system when a sensor which always predicts that no attack has occurred is added to the system. The upper bound is computed by choosing a selection of compromised sensors using a heuristic. For a step-wise increasing number of compromised sensors, the used heuristic selects the next sensor based on which sensor reduces the detection rate the most. If no sensor reduces the detection rate at the current number of compromised sensors, the sensor with the lowest PCFA is chosen. We use a heuristic instead of an exhaustive search as this would be infeasible for larger numbers of compromised sensors. It is apparent that this heuristic performs well in practice and leads to fairly tight lower and upper bounds.

One can see that all three fusion approaches generally outperform the monolithic classifier when it comes to the number of compromised sensors they are able to tolerate. The only event for which the previous fusion approach exhibits lower performance under compromise than the monolithic classifier is the “pc on” event. However, both the context-free and the context-aware extensions can still match the SVM’s performance in this case. Overall, the fusion approach from the previous chapter has an equal or higher tolerance to compromised sensors than the SVM classifier but tends to have lower detection rates with less compromised sensors. The SVM classifier shows high detection rates without compromise (except for the “window closed” event) but the detection rates quickly go to zero as more sensors are compromised. The performance of the context-aware extension matches that of the context-free sensor selection closely and exceeds it in three out of the six

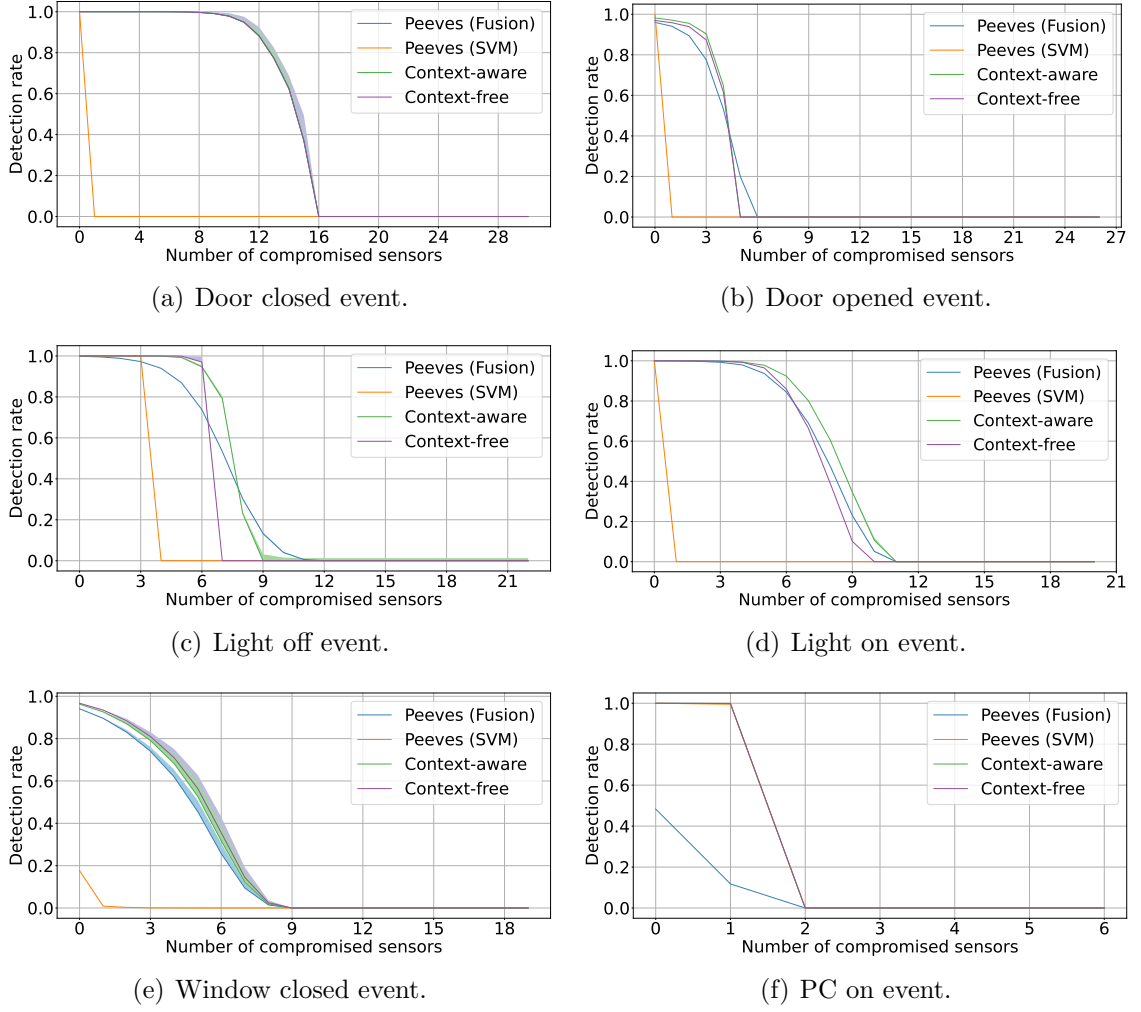


Figure 6.7: System performance against a sensor-compromise attacker with C sensors under their control for various events. Detection rates are given for the PEEVES monolithic classifier and the fusion approach from Chapter 5, the context-free extended sensor selection, and the context-aware extension for different numbers of C and with a false alarm rate of zero. For each of the three fusion-based approaches upper and lower bounds are given to make computation tractable. The lower bound is given as a solid line whereas the upper bound is shown as shaded area. More details can be found in the text.

shown events. Both approaches build on the higher robustness of the previous approach but improve on the performance for a smaller number of compromised sensors and when no sensor is compromised—often bringing the detection rates on par with or exceeding those of the SVM classifier.

6.5.3 Context Selection and Context Manipulation Attacks

As the use of contextual information for attack detection adds another potential attack vector, it is important to consider what additional capabilities the attacker might gain from manipulating contextual events or choosing to attack in specific contexts. Tables 6.1 and 6.2 show the detection rates of the context-aware extensions at false alarm rate zero for the three attacker types in Section 6.3. In both tables, cells are left empty for the context-aware attackers if they can not improve on the performance of the zero-effort attacker. Context-selective attackers and context-manipulating attackers are considered separately to highlight the differences between the two.

The detection rates against a zero-effort spoofing attacker are very high, with 17 out of 22 events achieving detection rates higher than 90% and just two events with detection rates below 50%. Masking attacks are much harder to detect due to the class imbalance and its implication on the detection rate at false alarm rate zero (*cf.*, Chapter 5 for more details), with only 9 out of 22 events achieving detection rates of over 90% and nine events with a detection rate below 50%.

For spoofing attacks, context selection attacks prove to be the most successful, as the attacker is able to reduce the detection rates for half of the events with six events exhibiting reductions of more than 5%. Similar to the opportunistic attacker from Chapter 5, this means, however, that the attacker has to wait until the appropriate context state has been reached before launching their attack, which can possibly lead to a long waiting time. If the attacker can not afford this waiting period required to choose the context, but is instead able to manipulate contextual event information, they can still reduce attack performance for 9 events. However, the achieved detection rates are rarely lower than those of a context-selective attacker and only exceed a 5% reduction twice compared to a zero-effort attacker.

For masking attacks, the advanced attacker types are less successful, with the context-selective attacker only being able to reduce detection rates in five cases, whereas the context-manipulating outperforms the zero-effort attacker only once. Yet it is important to remember that this is largely due to the already low detection

Spoofed event	Zero-effort attack	Context selection attack	Context manipulation attack
Door opened	98.20	94.13	94.80 (1)
Door closed	100.00		
Fridge opened	100.00		
Fridge closed	100.00		99.96 (2)
Light on	100.00		
Light off	100.00		99.99 (1)
Fan on	96.90	92.18	95.70 (1)
Fan off	98.56	98.50	
Shade up	99.16	99.16	
Shade down	96.25	96.25	
Coffee	95.22	94.05	94.52 (2)
PC on	100.00	100.00	
PC off	91.04	22.64	22.64 (1)
Screen on	99.98		
Screen off	56.47	53.02	53.02 (1)
Privacy on	77.33	45.09	
Privacy off	95.57	71.42	
Window opened	88.09	69.79	86.42 (2)
Window closed	96.29	90.66	
Radiator on	99.79		
Radiator off	29.07	20.39	20.39 (1)
Doorbell	0.00		

Table 6.1: Spoofing attack against the context-aware extension. Detection rates of the system are given for all three attacker types from Section 6.3. The number of manipulated context events are given in parentheses for the context-manipulating attacker. Manipulated contexts can be attacked through both spoofing or masking. Empty cells signify that the corresponding attacker could not improve on the zero-effort attacker for this event.

Masked event	Zero-effort attack	Context selection attack	Context manipulation attack
Door opened	87.76	85.57	
Door closed	100.00		
Fridge opened	100.00		
Fridge closed	100.00		
Light on	100.00		
Light off	100.00		
Fan on	100.00		
Fan off	0.00		
Shade up	91.05	90.00	
Shade down	10.61	0.00	0.00 (1)
Coffee	91.67	86.37	
PC on	100.00		
PC off	60.71	60.71	
Screen on	56.30		
Screen off	0.00		
Privacy on	0.00		
Privacy off	0.00		
Window opened	45.00		
Window closed	16.67		
Radiator on	89.47		
Radiator off	0.00		
Doorbell	13.33	0.00	

Table 6.2: Masking attack against the context-aware extension. Detection rates of the system are given for all three attacker types from Section 6.3. The number of manipulated context events are given in parentheses for the context-manipulating attacker. Manipulated contexts can be attacked through both spoofing or masking. Empty cells signify that the attack could not improve on the zero-effort attacker for this event.

performance against zero-effort masking attacks. Recall from Chapter 5 that one of the reasons for this difference between spoofing and masking detection performance is the strictness of the metric that we use here: detection rate at false alarm rate zero. Due to the flipped class imbalance, masking detection has to deal with a much larger source for false alarms—leading to a higher likelihood that the detection rate has to be reduced to account for them.

Context-manipulating attackers never benefit from spoofing or masking more than two events. The risk of getting detected quickly increases as the event verification is also used for contextual events. Hence, the benefit of attacking in a manipulated context is often outweighed by the added risk of detection during the manipulation.

For the event verification system, there is a clear trade-off between using other device states as context to improve performance without making the system more attackable. In our evaluation, we found that the context-manipulating attacker mainly leveraged two masked attacks on vulnerable events to launch their attack. This suggests that unverified or insufficiently verified events can be easily exploited if they are used to determine which sensors should be included in the sensor fusion. It is therefore important that contextual information is only used for context-aware event verification when it can be reliably verified itself, as the attacker can otherwise potentially bypass stronger sensors and sway the decision to their advantage.

6.6 Limitations

In this chapter, we have considered only a single context at a time for our context-aware sensor selection method. It is however possible that several contexts might affect a sensor in unison. For example, a light sensor might exhibit degraded performance only when the light is off and the window shade is down—but may remain unaffected if only one of these contexts occurs. A multi-context sensor selection method could take such phenomena into account, but this would increase the complexity of the system. Moreover, specific context combinations might not occur often enough to ensure significant representation in the training data.

Nevertheless, with enough training time, or if a specific combination of context states occurs more frequently, a limited amount of multi-context information could be supported by the system.

More broadly, a lack of sufficient training data is an issue that these types of systems generally have to contend with. Events in smart homes happen organically and can therefore occur relatively rarely. In particular, events that are affected by seasonal effects (*e.g.*, heating in the winter or air-conditioning in the summer) mean that the system might need training data for a full year in order to cover relevant cases. Although manually performing events repeatedly with the sole purpose of shortening training times may seem to be a solution, this should be avoided as it would lead to a very limited number of contexts to which the system is adapted, as well as artificial and unrealistic sensor signatures. To combat these issues, systems need to either tolerate longer training times, resort to periodic retraining phases, or bootstrap the training process with knowledge from external sources. Possible external sources of information could be provided by expert knowledge (*e.g.*, studies such as ours) or machine learning models transferred from other deployment settings. The use of expert knowledge has the potential to prove useful for (context-aware) sensor selection methods in particular, as these are based on physical phenomena that are likely to occur across different deployments. The transferability of machine learning models on the basis of smart home event sensor signatures is yet to be explored, but presents an interesting avenue of future research.

6.7 Conclusion

In this chapter, we have presented an extension to event verification systems for smart homes that allows these systems to consider context-dependent performance when selecting sensors for sensor fusion. We show that adding a more stringent sensor selection to the fusion scheme from Chapter 5 improves spoofing attack detection for 19 out of 22 events. Furthermore, we demonstrate that extending the sensor selection to take context-awareness into account further improves the detection performance for 7 of these 19 events.

The use of context-dependent sensor selection is not limited to event verification systems, but has the potential to also greatly benefit other security systems that leverage ubiquitous sensors in smart home settings. Due to the availability of contextual information in smart home systems, similar performance improvements can be expected for other use cases (*e.g.*, human presence recognition systems [5]).

However, we show that the use of this context information relies on the trustworthiness of the event sources providing the contextual knowledge—highlighting the need for a reliable event verification system. It is crucial that contextual events are themselves verified to avoid that an attacker can exploit these events to influence the sensor selection to their advantage.

We demonstrate that PEEVES, although exhibiting weaker performance with regards to masking attacks, shows high detection performance of spoofing attacks—making it very difficult for an attacker to exploit the increased attack surface. Indeed, an attacker that can selectively attack in vulnerable contexts is more successful than an attacker that is able to manipulate contextual information. Selecting specific contexts to attack, however, comes at the cost of increased waiting times for the attacker.

This chapter has shown that context-aware sensor selection increases the robustness of systems to context-dependent factors influencing system performance, and thus constitutes an important step to securing smart home environments through the use of ubiquitous sensors.

7

Conclusion

Contents

7.1	Summary of Results	167
7.2	Future Work	169
7.3	Final Remarks	170

7.1 Summary of Results

Over the course of this thesis, we have shown, with the aid of two examples, that the ubiquitous availability of sensors in smart homes can be used to enhance the security of smart home systems.

First, we have developed a drone detection system that relies only on the availability of radio signal strength information and can therefore easily be deployed on existing consumer hardware. Second, we have proposed a system for event verification in event-driven smart homes that can verify the occurrence of physical events through event signatures provided by the multitude of sensors already present in many smart home devices. We have demonstrated that such a system can effectively leverage the event information available in modern smart homes in order to independently adapt to its environment.

We have designed both of these systems in such a way that they provide additional security without inconveniencing the user—neither system requires extensive, manual system configuration or frequently raises false alarms. These systems can be deployed on smart home devices already distributed within the home, often requiring only a software update to expose the necessary sensor data at a sufficient fidelity, and are able to leverage multiple devices in tandem in order to better detect attacks.

We have presented two types of sophisticated attackers and have demonstrated their implications on the design of security systems based on ubiquitous sensors. We have introduced the notion of an opportunistic attacker—a novel adversary who can wait for and predict attack opportunities. While this type of attacker can provide a significant threat to these types of systems, we have shown that they often have to face long waiting times for appropriate opportunities to arise, and that these waiting times get longer the higher the detection rate is. The second advanced adversary that we discussed is an attacker who can compromise individual sensors that are part of the security system. We have demonstrated that systems that rely on a single monolithic classifier are vulnerable to this type of attacker, as even a single compromised sensor can generally be enough to defeat the system. We have shown that approaches based on fusing the decisions of classifiers trained on the data of individual sensors are more robust to this type of attacker.

Furthermore, we have shown that the usefulness of specific sensors can depend on the current context and overall state of the smart home. We have demonstrated how security systems can use smart home event information to discover some of these contexts autonomously and how they can use contextual event information for context-aware sensor selection to improve system performance. We have discussed that this selection method is preferably used with sensor fusion for practical reasons, as monolithic classifiers would require retraining for every discovered context. We have shown that the use of contextual information requires that the event sources providing the information are trustworthy. This highlights the importance of a reliable event verification system, such as the one introduced in this thesis.

7.2 Future Work

One of the downsides of ubiquitous sensors in home environments is their potential impact on user privacy and security. As part of the work on this thesis, we have observed that some sensors were able to perceive events in unexpected ways. For example, atmospheric sensors inside the fridge were able to detect events outside the fridge such as the opening of the office door, whereas power meters were able to detect switching events from other devices within the same room. This has implications beyond the attacks described in this thesis, as this unexpected information gain opens up new vectors for attacks (such as accessing sensitive information through sensors/devices believed to be innocuous). Furthermore, we have noticed that there were several reoccurring event sequences. Future work could look at how an opportunistic attacker might use these event sequences to predict behavior patterns based on a stream of events in order to increase their effectiveness by choosing the timing of the attack based on the states of other devices or co-occurring events.

While posing a risk to the privacy of users, interactions between a user and smart home devices could also be used to further enhance access control systems in smart homes. We expect that these interactions might produce user-specific sensor signatures. If proven to exist, these unique signatures could be used to support the identification and authentication of users in smart home environments.

In order to shorten the training time of self-learning systems based on ubiquitous sensing, future research could investigate the location-dependence of sensor signatures for specific event types or augment them with expert knowledge. If some events have a similar sensor signature across different event locations, this transferability could be used to bootstrap the training phase. This would reduce the risk of not having enough information during training due to rare events or contexts, as the knowledge gained in other locations could benefit new deployments.

Finally, it is well established that sensor signatures can change over time due to environmental changes. Future work might productively consider if systems like the ones described in this thesis will have to periodically retrain in order to keep performance levels acceptable over longer spans of time. If event information

is available, these systems will potentially be able to relearn independently. For example, an event verification system can stay current by repeating the initial learning process, and a drone detection system could autonomously improve by leveraging nearby transmitters with known MAC address prefixes. When pursuing this line of research, however, it is important to consider the capabilities of the attackers discussed in this thesis, as an attacker might be able to exploit the retraining process, poison the training data and thus overcome the security of the system.

7.3 Final Remarks

With the rising popularity of smart home systems, IoT devices have become increasingly prevalent in peoples' homes and with them has come the availability of a large number of networked sensors. Although the growing pervasiveness of IoT devices and their sensors is often considered to be a privacy risk, it also opens up opportunities to strengthen the security of home environments due to increased situational awareness.

In this thesis, we have demonstrated that the ubiquity of sensors in smart homes can be used to enhance the security of smart home systems (such as event-driven smart homes), as well as to add new security features (such as drone detection). As more and more aspects of daily life are “smartified”, we expect that these principles will be applied to new use cases and that many of the lessons learned in this thesis will be transferable. By combining fine-grained sensor data with high-level event information, ubiquitous sensing will be able to provide stronger security properties in future smart home systems—without impinging on the smart devices' original functions.

References

- [1] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. “DDoS in the IoT: Mirai and Other Botnets”. In: *Computer* 50.7 (2017), pp. 80–84.
- [2] Markus Miettinen, N. Asokan, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and Majid Sobhani. “Context-Based Zero-Interaction Pairing and Key Evolution for Advanced Personal Devices”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. CCS ’14*. New York, NY, USA: ACM, 2014, 880–891.
- [3] Jun Han, Albert Jin Chung, Manal Kumar Sinha, Madhumitha Harishankar, Shijia Pan, Hae Young Noh, Pei Zhang, and Patrick Tague. “Do You Feel What I Hear? Enabling Autonomous IoT Device Pairing Using Different Sensor Types”. In: *2018 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, pp. 836–852.
- [4] Joey Wilson and Neal Patwari. “Radio tomographic imaging with wireless networks”. In: *IEEE Transactions on Mobile Computing* 9.5 (2010), pp. 621–632.
- [5] Thiago Teixeira, Gershon Dublon, and Andreas Savvides. “A survey of human-sensing: Methods for detecting presence, count, location, track, and identity”. In: *ACM Computing Surveys* 5.1 (2010), pp. 59–69.
- [6] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z Morley Mao, and Atul Prakash. “ContextIoT: Towards Providing Contextual Integrity to Appified IoT Platforms”. In: *Proceedings of the 24th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, California, USA: The Internet Society, 2017.
- [7] Amir Rahmati, Earlence Fernandes, Kevin Eykholt, and Atul Prakash. “Tyche: A Risk-Based Permission Model for Smart Homes”. In: *2018 IEEE Cybersecurity Development (SecDev)*. IEEE, 2018, pp. 29–36.
- [8] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. “FlowFence: Practical Data Protection for Emerging IoT Application Frameworks”. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 531–548.
- [9] Sandra Siby, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. “IoTScanner: Detecting Privacy Threats in IoT Neighborhoods”. In: *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security. IoTPTS ’17*. New York, NY, USA: ACM, 2017, pp. 23–30.

- [10] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. “IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT”. In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184.
- [11] Shahid Raza, Linus Wallgren, and Thiemo Voigt. “SVELTE: Real-time intrusion detection in the Internet of Things”. In: *Ad Hoc Networks* 11.8 (2013), pp. 2661–2674.
- [12] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. “Is Anybody Home? Inferring Activity From Smart Home Network Traffic”. In: *2016 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2016, pp. 245–251.
- [13] Simon Birnbach, Richard Baker, and Ivan Martinovic. “Wi-Fly?: Detecting Privacy Invasion Attacks by Consumer Drones”. In: *Proceedings of the 24th Annual Network and Distributed System Security Symposium (NDSS)*. 2017.
- [14] Simon Birnbach, Richard Baker, Simon Eberz, and Ivan Martinovic. “#PrettyFlyForAWiFi : Real-World Detection of Privacy Invasion Attacks by Drones”. In: *ACM Trans. Priv. Secur. (TOPS)* 24.4 (Sept. 2021). URL: <https://doi.org/10.1145/3473672>.
- [15] Simon Birnbach, Simon Eberz, and Ivan Martinovic. “Peeves: Physical Event Verification in Smart Homes”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom). CCS ’19. New York, NY, USA: Association for Computing Machinery, 2019, 1455–1467. URL: <https://doi.org/10.1145/3319535.3354254>.
- [16] Simon Birnbach, Simon Eberz, and Ivan Martinovic. “Haunted House: Physical Smart Home Event Verification in the Presence of Compromised Sensors”. In: *ACM Trans. Internet Things (TIOT)* 3.3 (2022). URL: <https://doi.org/10.1145/3506859>.
- [17] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. “Security Analysis of Emerging Smart Home Applications”. In: *2016 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2016, pp. 636–654.
- [18] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. “SmartAuth: User-Centered Authorization for the Internet of Things”. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 361–378.
- [19] Sanghak Lee, Jiwon Choi, Jihun Kim, Beumjin Cho, Sangho Lee, Hanjun Kim, and Jong Kim. “FACT: Functionality-Centric Access Control System for IoT Programming Frameworks”. In: *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies* (Indianapolis, Indiana, USA). SACMAT ’17 Abstracts. New York, NY, USA: Association for Computing Machinery, 2017, 43–54. URL: <https://doi.org/10.1145/3078861.3078864>.

- [20] Moosa Yahyazadeh, Proyash Podder, Endadul Hoque, and Omar Chowdhury. “Expat: Expectation-Based Policy Analysis and Enforcement for Appified Smart-Home Platforms”. In: *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies* (Toronto ON, Canada). SACMAT '19. New York, NY, USA: Association for Computing Machinery, 2019, 61–72. URL: <https://doi.org/10.1145/3322431.3325107>.
- [21] Yin hao Xiao, Yizhen Jia, Chunchi Liu, Arwa Alrawais, Molka Rekik, and Zhiguang Shan. “HomeShield: A Credential-Less Authentication Framework for Smart Home Systems”. In: *IEEE Internet of Things Journal* 7.9 (2020), pp. 7903–7918.
- [22] Z. Berkay Celik, Gang Tan, and Patrick D. McDaniel. “IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT”. In: *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, California, USA: The Internet Society, 2019.
- [23] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Jiaping Yu. “Cross-App Interference Threats in Smart Homes: Categorization, Detection and Handling”. In: *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2020, pp. 411–423.
- [24] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. “Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things”. In: *Proceedings of the 14th ACM Workshop on Hot Topics in Networks* (Philadelphia, PA, USA). HotNets-XIV. New York, NY, USA: Association for Computing Machinery, 2015. URL: <https://doi.org/10.1145/2834050.2834095>.
- [25] Leonardo Babun, Z Berkay Celik, Patrick McDaniel, and A Selcuk Uluagac. “Real-time analysis of privacy-(un)aware IoT applications”. In: *Proceedings on Privacy Enhancing Technologies* 2021.1 (2021), pp. 145–166.
- [26] Sanket Goutam, William Enck, and Bradley Reaves. “Hestia: Simple Least Privilege Network Policies for Smart Homes”. In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks* (Miami, Florida). WiSec '19. New York, NY, USA: Association for Computing Machinery, 2019, 215–220. URL: <https://doi.org/10.1145/3317549.3323413>.
- [27] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. “Sensitive Information Tracking in Commodity IoT”. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1687–1704. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/celik>.
- [28] Z. Berkay Celik, Patrick McDaniel, and Gang Tan. “Soteria: Automated IoT Safety and Security Analysis”. In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, 2018, pp. 147–158.
- [29] Wenbo Ding and Hongxin Hu. “On the Safety of IoT Device Physical Interaction Control”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. New York, NY, USA: ACM, 2018, 832–846.

- [30] Dang Tu Nguyen, Chengyu Song, Zhiyun Qian, Srikanth V. Krishnamurthy, Edward J. M. Colbert, and Patrick McDaniel. “IoTSan: Fortifying the Safety of IoT Systems”. In: *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT ’18. New York, NY, USA: ACM, 2018, 191–203.
- [31] Kai-Hsiang Hsu, Yu-Hsi Chiang, and Hsu-Chun Hsiao. “SafeChain: Securing Trigger-Action Programming From Attack Chains”. In: *IEEE Transactions on Information Forensics and Security* 14.10 (2019), pp. 2607–2622.
- [32] Musard Balliu, Massimo Merro, Michele Pasqua, and Mikhail Shcherbakov. “Friendly Fire: Cross-App Interactions in IoT Platforms”. In: *ACM Trans. Priv. Secur.* 24.3 (Apr. 2021). URL: <https://doi.org/10.1145/3444963>.
- [33] Mujahid Mohsin, Zahid Anwar, Ghaith Husari, Ehab Al-Shaer, and Mohammad Ashiqur Rahman. “IoTSAT: A formal framework for security analysis of the internet of things (IoT)”. In: *2016 IEEE Conference on Communications and Network Security (CNS)*. 2016, pp. 180–188.
- [34] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. “Charting the Attack Surface of Trigger-Action IoT Platforms”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom). CCS ’19. New York, NY, USA: Association for Computing Machinery, 2019, 1439–1453. URL: <https://doi.org/10.1145/3319535.3345662>.
- [35] Wenbo Ding, Hongxin Hu, and Long Cheng. “IOTS SAFE: Enforcing Safety and Security Policy with Real IoT Physical Interaction Discovery”. In: *Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, California, USA: The Internet Society, 2021. URL: <https://www.ndss-symposium.org/ndss-paper/iotsafe-enforcing-safety-and-security-policy-with-real-iot-physical-interaction-discovery/>.
- [36] Milijana Surbatovich, Jassim Aljuraidan, Lujo Bauer, Anupam Das, and Limin Jia. “Some Recipes Can Do More Than Spoil Your Appetite: Analyzing the Security and Privacy Risks of IFTTT Recipes”. In: *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia). WWW ’17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, 1501–1510. URL: <https://doi.org/10.1145/3038912.3052709>.
- [37] Earlene Fernandes, Amir Rahmati, Jaeyeon Jung, and Atul Prakash. “Decentralized action integrity for trigger-action IoT platforms”. In: *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, California, USA: The Internet Society, 2018.
- [38] Iulia Bastys, Musard Balliu, and Andrei Sabelfeld. “If This Then What? Controlling Flows in IoT Apps”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada). CCS ’18. New York, NY, USA: Association for Computing Machinery, 2018, 1102–1119. URL: <https://doi.org/10.1145/3243734.3243841>.

- [39] Camille Cobb, Milijana Surbatovich, Anna Kawakami, Mahmood Sharif, Lujo Bauer, Anupam Das, and Limin Jia. “How Risky Are Real Users’ IFTTT Applets?” In: *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, Aug. 2020, pp. 505–529. URL: <https://www.usenix.org/conference/soups2020/presentation/cobb>.
- [40] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. “Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms”. In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1133–1150. URL: <https://www.usenix.org/conference/usenixsecurity19/presentation/zhou>.
- [41] Jiongyi Chen, Chaoshun Zuo, Wenrui Diao, Shuaike Dong, Qingchuan Zhao, Menghan Sun, Zhiqiang Lin, Yinqian Zhang, and Kehuan Zhang. “Your IoTs Are (Not) Mine: On the Remote Binding Between IoT Devices and Users”. In: *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2019, pp. 222–233.
- [42] Yan Jia, Luyi Xing, Yuhang Mao, Dongfang Zhao, XiaoFeng Wang, Shangru Zhao, and Yuqing Zhang. “Burglars’ IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds”. In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 465–481.
- [43] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. “Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic”. In: (2017). arXiv: 1708.05044 [cs.CR].
- [44] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. “Keeping the smart home private with smart(er) iot traffic shaping”. In: *Proceedings on Privacy Enhancing Technologies 2019.3* (2019), pp. 128–148.
- [45] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. “Peek-a-Boo: I See Your Smart Home Activities, Even Encrypted!” In: *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec ’20. New York, NY, USA: ACM, 2020, 207–218.
- [46] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. “Is Anybody Home? Inferring Activity From Smart Home Network Traffic”. In: *2016 IEEE Security and Privacy Workshops (SPW)*. 2016, pp. 245–251.
- [47] Rixin Xu, Qiang Zeng, Liehuang Zhu, Haotian Chi, Xiaojiang Du, and Mohsen Guizani. “Privacy Leakage in Smart Homes and Its Mitigation: IFTTT as a Case Study”. In: *IEEE Access* 7 (2019), pp. 63457–63471.
- [48] Eric Zeng, Shrirang Mare, and Franziska Roesner. “End User Security and Privacy Concerns with Smart Homes”. In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, July 2017, pp. 65–80. URL: <https://www.usenix.org/conference/soups2017/technical-sessions/presentation/zeng>.

- [49] Tamara Denning, Tadayoshi Kohno, and Henry M. Levy. “Computer Security and the Modern Home”. In: *Communications of the ACM* 56.1 (2013), pp. 94–103.
- [50] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. “Smart Locks: Lessons for Securing Commodity Internet of Things Devices”. In: *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ASIA CCS ’16. New York, NY, USA: ACM, 2016, pp. 461–472.
- [51] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O’Flynn. “IoT Goes Nuclear: Creating a ZigBee Chain Reaction”. In: *2017 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 195–212.
- [52] Blase Ur, Jaeyeon Jung, and Stuart Schechter. “The Current State of Access Control for Smart Devices in Homes”. In: *Workshop on Home Usable Privacy and Security (HUPS)*. Vol. 29. HUPS 2014. July 2013, pp. 209–218.
- [53] Jens Müller, Vladislav Mladenov, Juraj Somorovsky, and Jörg Schwenk. “SoK: Exploiting Network Printers”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 213–230.
- [54] Zhen Ling, Junzhou Luo, Yiling Xu, Chao Gao, Kui Wu, and Xinwen Fu. “Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System”. In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1899–1909.
- [55] Sukhvir Notra, Muhammad Siddiqi, Hassan Habibi Gharakheili, Vijay Sivaraman, and Roksana Boreli. “An experimental study of security and privacy risks with emerging household appliances”. In: *2014 IEEE Conference on Communications and Network Security*. 2014, pp. 79–84.
- [56] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. “Understanding the Mirai Botnet”. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>.
- [57] Harm Griffioen and Christian Doerr. “Examining Mirai’s Battle over the Internet of Things”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2020, 743–756. URL: <https://doi.org/10.1145/3372297.3417277>.
- [58] Sam Edwards and Ioannis Profetis. “Hajime: Analysis of a decentralized internet worm for IoT devices”. In: *Rapidity Networks* 16 (2016), pp. 1–18.
- [59] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. “Measurement and analysis of Hajime, a peer-to-peer IoT botnet”. In: *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, California, USA: The Internet Society, 2019.

- [60] Tim Yeh, Dove Chiu, and Kenney Lu. *Persirai: New IoT botnet targets IP cameras*. Accessed: 10.02.2021. 2017. URL: https://www.trendmicro.com/en_gb/research/17/e/persirai-new-internet-things-iot-botnet-targets-ip-cameras.html.
- [61] Radware. *"BrickerBot" Results In Permanent Denial-of-Service*. Accessed: 10.02.2021. 2017. URL: <https://www.radware.com/security/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/>.
- [62] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. "Analysis of DDoS-Capable IoT Malwares". In: *Proceedings of the 1st International Conference on Security, Privacy, and Trust (INSERT)*, IEEE. 2017.
- [63] Saleh Soltan, Prateek Mittal, and H. Vincent Poor. "BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid". In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 15–32. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/soltan>.
- [64] Vijay Sivaraman, Dominic Chan, Dylan Earl, and Roksana Boreli. "Smart-Phones Attacking Smart-Homes". In: *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '16. New York, NY, USA: ACM, 2016, pp. 195–200.
- [65] Shodan. *The search engine for the internet of things*. Accessed: 18.06.2021. 2021. URL: <https://www.shodan.io>.
- [66] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. "Behavioral Fingerprinting of IoT Devices". In: *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*. ASHES '18. New York, NY, USA: ACM, 2018, 41–50.
- [67] Samuel Marchal, Markus Miettinen, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and N. Asokan. "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication". In: *IEEE Journal on Selected Areas in Communications* 37.6 (2019), pp. 1402–1412.
- [68] Xuan Feng, Qiang Li, Haining Wang, and Limin Sun. "Acquisitional Rule-based Engine for Discovering Internet-of-Things Devices". In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 327–341. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/feng>.
- [69] Juan Wang, Shirong Hao, Ru Wen, Boxian Zhang, Liqiang Zhang, Hongxin Hu, and Rongxing Lu. "IoT-Praetor: Undesired Behaviors Detection for IoT Devices". In: *IEEE Internet of Things Journal* 8.2 (2021), pp. 927–940.
- [70] TJ OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. "HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devices". In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks* (Miami, Florida). WiSec '19. New York, NY, USA: Association for Computing Machinery, 2019, 128–138. URL: <https://doi.org/10.1145/3317549.3323409>.

- [71] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. “HoMonit: Monitoring Smart Home Apps from Encrypted Traffic”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. New York, NY, USA: ACM, 2018, 1074–1088.
- [72] Tianbo Gu, Zheng Fang, Allaukik Abhishek, Hao Fu, Pengfei Hu, and Prasant Mohapatra. “IoTGaze: IoT Security Enforcement via Wireless Context Analysis”. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications* (Toronto, ON, Canada). IEEE Press, 2020, 884–893. URL: <https://doi.org/10.1109/INFOCOM41043.2020.9155459>.
- [73] Masaaki Yamauchi, Yuichi Ohsita, Masayuki Murata, Kensuke Ueda, and Yoshiaki Kato. “Anomaly Detection for Smart Home Based on User Behavior”. In: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. 2019, pp. 1–6.
- [74] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. “D²IoT: A Federated Self-learning Anomaly Detection System for IoT”. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 2019, pp. 756–767.
- [75] Amit Kumar Sikder, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. “Aegis: A Context-Aware Security Framework for Smart Home Systems”. In: *Proceedings of the 35th Annual Computer Security Applications Conference* (San Juan, Puerto Rico, USA). ACSAC ’19. New York, NY, USA: Association for Computing Machinery, 2019, 28–41. URL: <https://doi.org/10.1145/3359789.3359840>.
- [76] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. “HAWatcher: Semantics-Aware Anomaly Detection for Appified Smart Homes”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 4223–4240. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/fu-chenglong>.
- [77] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl A. Gunter. “Fear and Logging in the Internet of Things”. In: *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, California, USA: The Internet Society, 2018.
- [78] Leonardo Babun, Amit Kumar Sikder, Abbas Acar, and A Selcuk Uluagac. “Iotdots: A digital forensics framework for smart environments”. In: (2018). arXiv: 1809.00745 [cs.CR].
- [79] Shams Zawoad and Ragib Hasan. “FAIoT: Towards Building a Forensics Aware Eco System for the Internet of Things”. In: *2015 IEEE International Conference on Services Computing*. 2015, pp. 279–284.
- [80] Victor R. Kebande and Indrakshi Ray. “A Generic Digital Forensic Investigation Framework for Internet of Things (IoT)”. In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*. 2016, pp. 356–362.

- [81] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. “IoTPOT: Analysing the Rise of IoT Compromises”. In: *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. USENIX Association, 2015.
- [82] Michele De Donno, Nicola Dragoni, Alberto Giarretta, and Manuel Mazzara. “AntibIoTic: Protecting IoT Devices Against DDoS Attacks”. In: (2017). arXiv: 1708.05050 [cs.CR].
- [83] Mikhail Fomichev, Flor Alvarez, Daniel Steinmetzer, Paul Gardner-Stephen, and Matthias Hollick. “Survey and Systematization of Secure Device Pairing”. In: *IEEE Communications Surveys & Tutorials* (2017).
- [84] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal De Lara. “Amigo: Proximity-based authentication of mobile devices”. In: *UbiComp 2007: Ubiquitous Computing* (2007), pp. 253–270.
- [85] D. Schürmann and S. Sigg. “Secure Communication Based on Ambient Audio”. In: *IEEE Transactions on Mobile Computing* 12.2 (2013), pp. 358–370.
- [86] Andre Kalamandeen, Adin Scannell, Eyal de Lara, Anmol Sheth, and Anthony LaMarca. “Ensemble: cooperative proximity-based authentication”. In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM. 2010, pp. 331–344.
- [87] Suhas Mathur, Robert Miller, Alexander Varshavsky, Wade Trappe, and Narayan Mandayam. “ProxiMate: Proximity-Based Secure Pairing Using Ambient Wireless Signals”. In: *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services* (Bethesda, Maryland, USA). MobiSys ’11. New York, NY, USA: Association for Computing Machinery, 2011, 211–224. URL: <https://doi.org/10.1145/1999995.2000016>.
- [88] Tzipora Halevi, Di Ma, Nitesh Saxena, and Tuo Xiang. “Secure proximity detection for NFC devices based on ambient sensor data”. In: *European Symposium on Research in Computer Security*. Springer. 2012, pp. 379–396.
- [89] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N Asokan. “Drone to the rescue: Relay-resilient authentication using ambient multi-sensing”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 349–364.
- [90] Christian T. Zenger, Jan Zimmer, Mario Pietersz, Jan-Felix Posielek, and Christof Paar. “Exploiting the Physical Environment for Securing the Internet of Things”. In: *Proceedings of the 2015 New Security Paradigms Workshop* (Twente, Netherlands). NSPW ’15. New York, NY, USA: Association for Computing Machinery, 2015, 44–58. URL: <https://doi.org/10.1145/2841113.2841117>.
- [91] B. Shrestha, N. Saxena, H. T. T. Truong, and N. Asokan. “Sensor-based Proximity Detection in the Face of Active Adversaries”. In: *IEEE Transactions on Mobile Computing* (2018), pp. 1–1.
- [92] Kyuin Lee, Neil Klingensmith, Suman Banerjee, and Younghyun Kim. “VoltKey: Continuous Secret Key Generation Based on Power Line Noise for Zero-Involvement Pairing and Authentication”. In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3.3 (Sept. 2019). URL: <https://doi.org/10.1145/3351251>.

- [93] Kevin Fu and Wenyuan Xu. “Risks of trusting the physics of sensors”. In: *Communications of the ACM Magazine* 61.2 (2018), pp. 20–23.
- [94] Ilias Giechaskiel and Kasper Rasmussen. “Taxonomy and Challenges of Out-of-Band Signal Injection Attacks and Defenses”. In: *IEEE Communications Surveys & Tutorials* 22.1 (2020), pp. 645–670.
- [95] Youngseok Park, Yunmok Son, Hocheol Shin, Dohyun Kim, and Yongdae Kim. “This Ain’t Your Dose: Sensor Spoofing Attack on Medical Infusion Pump”. In: *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, Aug. 2016. URL: <https://www.usenix.org/conference/woot16/workshop-program/presentation/park>.
- [96] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. “Non-invasive spoofing attacks for anti-lock braking systems”. In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2013, pp. 55–72.
- [97] Connor Bolton, Sara Rampazzi, Chaohao Li, Andrew Kwong, Wenyuan Xu, and Kevin Fu. “Blue Note: How Intentional Acoustic Interference Damages Availability and Integrity in Hard Disk Drives and Operating Systems”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 1048–1062.
- [98] Drew Davidson, Hao Wu, Rob Jellinek, Vikas Singh, and Thomas Ristenpart. “Controlling UAVs with Sensor Input Spoofing Attacks”. In: *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, Aug. 2016. URL: <https://www.usenix.org/conference/woot16/workshop-program/presentation/davidson>.
- [99] Jayaprakash Selvaraj, Gökçen Yılmaz Dayanıklı, Neelam Prabhu Gaunkar, David Ware, Ryan M. Gerdes, and Mani Mina. “Electromagnetic Induction Attacks Against Embedded Systems”. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (Incheon, Republic of Korea). ASIACCS ’18. New York, NY, USA: Association for Computing Machinery, 2018, 499–510. URL: <https://doi.org/10.1145/3196494.3196556>.
- [100] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. “Rocking drones with intentional sound noise on gyroscopic sensors”. In: *24th USENIX Security Symposium (USENIX Security 15)*. 2015, pp. 881–896.
- [101] Yazhou Tu, Zhiqiang Lin, Insup Lee, and Xiali Hei. “Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors”. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1545–1562. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/tu>.
- [102] Shoei Nashimoto, Daisuke Suzuki, Takeshi Sugawara, and Kazuo Sakiyama. “Sensor CON-Fusion: Defeating Kalman Filter in Signal Injection Attack”. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (Incheon, Republic of Korea). ASIACCS ’18. New York, NY, USA: Association for Computing Machinery, 2018, 511–524. URL: <https://doi.org/10.1145/3196494.3196506>.

- [103] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. “WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks”. In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2017, pp. 3–18.
- [104] Denis Foo Kune, John Backes, Shane S Clark, Daniel Kramer, Matthew Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. “Ghost talk: Mitigating EMI signal injection attacks against analog sensors”. In: *2013 IEEE Symposium on Security and Privacy (S&P)*. IEEE. 2013, pp. 145–159.
- [105] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. “DolphinAttack: Inaudible voice commands”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM. 2017, pp. 103–117.
- [106] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. “Inaudible Voice Commands: The Long-Range Attack and Defense”. In: *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 547–560. URL: <https://www.usenix.org/conference/nsdi18/presentation/roy>.
- [107] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. “BackDoor: Making Microphones Hear Inaudible Sounds”. In: *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services* (Niagara Falls, New York, USA). MobiSys ’17. New York, NY, USA: Association for Computing Machinery, 2017, 2–14. URL: <https://doi.org/10.1145/3081333.3081366>.
- [108] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. “Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems”. In: *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2631–2648. URL: <https://www.usenix.org/conference/usenixsecurity20/presentation/sugawara>.
- [109] Ilias Giechaskiel, Youqian Zhang, and Kasper B Rasmussen. “A framework for evaluating security in the presence of signal injection attacks”. In: *European Symposium on Research in Computer Security*. Springer. 2019, pp. 512–532.
- [110] Yazhou Tu, Sara Rampazzi, Bin Hao, Angel Rodriguez, Kevin Fu, and Xiali Hei. “Trick or Heat? Manipulating Critical Temperature-Based Control Systems Using Rectification Attacks”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom). CCS ’19. New York, NY, USA: Association for Computing Machinery, 2019, 2301–2315. URL: <https://doi.org/10.1145/3319535.3354195>.
- [111] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. “Pycra: Physical challenge-response authentication for active sensors under spoofing attacks”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM. 2015, pp. 1004–1015.
- [112] Chuadhry Mujeeb Ahmed, Aditya Mathur, and Martin Ochoa. “NoiSense: Detecting Data Integrity Attacks on Sensor Measurements using Hardware based Fingerprints”. In: (2017). arXiv: 1712.01598 [cs.CR].

- [113] Hocheol Shin, Yunmok Son, Youngseok Park, Yujin Kwon, and Yongdae Kim. “Sampling race: Bypassing timing-based analog active sensor spoofing detection on analog-digital systems”. In: *10th USENIX Workshop on Offensive Technologies (WOOT)*. USENIX. 2016.
- [114] Devaprakash Muniraj and Mazen Farhood. “Detection and mitigation of actuator attacks on small unmanned aircraft systems”. In: *Control Engineering Practice* 83 (2019), pp. 188–202. URL: <https://www.sciencedirect.com/science/article/pii/S0967066118306804>.
- [115] Kevin Sam Tharayil, Benyamin Farshteindiker, Shaked Eyal, Nir Hasidim, Roy Hershkovitz, Shani Houri, Ilia Yoffe, Michal Oren, and Yossi Oren. “Sensor Defense In-Software (SDI): Practical software based detection of spoofing attacks on position sensors”. In: *Engineering Applications of Artificial Intelligence* 95 (2020), p. 103904. URL: <https://www.sciencedirect.com/science/article/pii/S0952197620302402>.
- [116] Youqian Zhang and Kasper Rasmussen. “Detection of Electromagnetic Interference Attacks on Sensor Systems”. In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 203–216.
- [117] Yao Liu, Peng Ning, and Michael K. Reiter. “False Data Injection Attacks against State Estimation in Electric Power Grids”. In: *ACM Trans. Inf. Syst. Secur.* 14.1 (June 2011). URL: <https://doi.org/10.1145/1952982.1952995>.
- [118] Miroslav Pajic, Insup Lee, and George J. Pappas. “Attack-Resilient State Estimation for Noisy Dynamical Systems”. In: *IEEE Transactions on Control of Network Systems* 4.1 (2017), pp. 82–92.
- [119] Sze Zheng Yong, Minghui Zhu, and Emilio Frazzoli. “Switching and Data Injection Attacks on Stochastic Cyber-Physical Systems: Modeling, Resilient Estimation, and Attack Mitigation”. In: *ACM Trans. Cyber-Phys. Syst.* 2.2 (June 2018). URL: <https://doi.org/10.1145/3204439>.
- [120] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. “Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks”. In: *IEEE Transactions on Automatic Control* 59.6 (2014), pp. 1454–1467.
- [121] Derui Ding, Qing-Long Han, Xiaohua Ge, and Jun Wang. “Secure State Estimation and Control of Cyber-Physical Systems: A Survey”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.1 (2021), pp. 176–190.
- [122] Junkil Park, Radoslav Ivanov, James Weimer, Miroslav Pajic, Sang Hyuk Son, and Insup Lee. “Security of Cyber-Physical Systems in the Presence of Transient Sensor Faults”. In: *ACM Trans. Cyber-Phys. Syst.* 1.3 (May 2017). URL: <https://doi.org/10.1145/3064809>.
- [123] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. “A Survey of Physics-Based Attack Detection in Cyber-Physical Systems”. In: *ACM Comput. Surv.* 51.4 (July 2018). URL: <https://doi.org/10.1145/3203245>.

- [124] Long Cheng, Ke Tian, and Danfeng Daphne Yao. “Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks”. In: *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC)*. ACM. 2017, pp. 315–326.
- [125] Alvaro A. Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. “Attacks against Process Control Systems: Risk Assessment, Detection, and Response”. In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (Hong Kong, China). ASIACCS '11. New York, NY, USA: Association for Computing Machinery, 2011, 355–366. URL: <https://doi.org/10.1145/1966913.1966959>.
- [126] Dina Hadžiosmanović, Robin Sommer, Emmanuele Zambon, and Pieter H. Hartel. “Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes”. In: *Proceedings of the 30th Annual Computer Security Applications Conference* (New Orleans, Louisiana, USA). ACSAC '14. New York, NY, USA: Association for Computing Machinery, 2014, 126–135. URL: <https://doi.org/10.1145/2664243.2664277>.
- [127] Chathurika S. Wickramasinghe, Daniel L. Marino, Kasun Amarasinghe, and Milos Manic. “Generalization of Deep Learning for Cyber-Physical System Security: A Survey”. In: *44th Annual Conference of the IEEE Industrial Electronics Society (IECON)*. 2018, pp. 745–751.
- [128] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. “Constrained Concealment Attacks against Reconstruction-Based Anomaly Detectors in Industrial Control Systems”. In: *Annual Computer Security Applications Conference* (Austin, USA). ACSAC '20. New York, NY, USA: Association for Computing Machinery, 2020, 480–495. URL: <https://doi.org/10.1145/3427228.3427660>.
- [129] Vittorio P. Illiano and Emil C. Lupu. “Detecting Malicious Data Injections in Wireless Sensor Networks: A Survey”. In: *ACM Comput. Surv.* 48.2 (Oct. 2015). URL: <https://doi.org/10.1145/2818184>.
- [130] F. Liu, X. Cheng, and D. Chen. “Insider Attacker Detection in Wireless Sensor Networks”. In: *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. 2007, pp. 1937–1945.
- [131] S. Tanachaiwiwat and A. Helmy. “Correlation analysis for alleviating effects of inserted data in wireless sensor networks”. In: *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. 2005, pp. 97–108.
- [132] Luís Bettencourt, Aric A Hagberg, and Levi B Larkey. “Separating the wheat from the chaff: Practical anomaly detection schemes in ecological applications of distributed sensor networks”. In: *International Conference on Distributed Computing in Sensor Systems*. Springer. 2007, pp. 223–239.
- [133] Sutharshan Rajasegarar, Christopher Leckie, Marimuthu Palaniswami, and James C. Bezdek. “Distributed Anomaly Detection in Wireless Sensor Networks”. In: *2006 10th IEEE Singapore International Conference on Communication Systems*. 2006, pp. 1–5.

- [134] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek. “Quarter Sphere Based Distributed Anomaly Detection in Wireless Sensor Networks”. In: *2007 IEEE International Conference on Communications*. 2007, pp. 3864–3869.
- [135] V. P. Illiano, L. Muñoz-González, and E. C. Lupu. “Don’t fool Me!: Detection, Characterisation and Diagnosis of Spoofed and Masked Events in Wireless Sensor Networks”. In: *IEEE Transactions on Dependable and Secure Computing* 14.3 (2017), pp. 279–293.
- [136] Wei Zhang, Sajal K. Das, and Yonghe Liu. “A Trust Based Framework for Secure Data Aggregation in Wireless Sensor Networks”. In: *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*. Vol. 1. 2006, pp. 60–69.
- [137] Fenyue Bao, Ing-Ray Chen, MoonJeong Chang, and Jin-Hee Cho. “Hierarchical Trust Management for Wireless Sensor Networks and its Applications to Trust-Based Routing and Intrusion Detection”. In: *IEEE Transactions on Network and Service Management* 9.2 (2012), pp. 169–183.
- [138] Idris M. Atakli, Hongbing Hu, Yu Chen, Wei Shinn Ku, and Zhou Su. “Malicious Node Detection in Wireless Sensor Networks Using Weighted Trust Evaluation”. In: *Proceedings of the 2008 Spring Simulation Multiconference* (Ottawa, Canada). SpringSim ’08. San Diego, CA, USA: Society for Computer Simulation International, 2008, 836–843.
- [139] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici. “SoK: Security and Privacy in the Age of Commercial Drones”. In: *2021 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, 2021, pp. 73–90.
- [140] Ellen E Case, Anne M Zelnio, and Brian D Rigling. “Low-Cost Acoustic Array for Small UAV Detection and Tracking”. In: *2008 IEEE National Aerospace and Electronics Conference*. IEEE. 2008.
- [141] Minas Benyamin and Geoffrey H Goldman. *Acoustic detection and tracking of a Class I UAS with a small tetrahedral microphone array*. Tech. rep. Army Research Lab, 2014.
- [142] Joël Busset, Florian Perrodin, Peter Wellig, Beat Ott, Kurt Heutschi, Torben Rühl, and Thomas Nussbaumer. “Detection and tracking of drones using advanced acoustic cameras”. In: *SPIE Security+ Defence*. International Society for Optics and Photonics. 2015.
- [143] Juan R Vasquez, Kyle M Tarplee, Ellen E Case, Anne M Zelnio, and Brian D Rigling. “Multisensor 3D tracking for counter small unmanned air vehicles (CSUAV)”. In: *SPIE Defense and Security Symposium*. International Society for Optics and Photonics. 2008.
- [144] X. Shi, C. Yang, W. Xie, C. Liang, Z. Shi, and J. Chen. “Anti-Drone System with Multiple Surveillance Technologies: Architecture, Implementation, and Challenges”. In: *IEEE Communications Magazine* 56.4 (2018), pp. 68–74.
- [145] Gabriel C Birch and Bryana L Woo. “Counter Unmanned Aerial Systems Testing: Evaluation of VIS SWIR MWIR and LWIR Passive Imagers”. In: *SANDIA REPORT SAND2017-0921* (2017).

- [146] Philip Church, Christopher Grebe, Justin Matheson, and Brett Owens. “Aerial and surface security applications using lidar”. In: *Laser Radar Technology and Applications XXIII*. Vol. 10636. International Society for Optics and Photonics. 2018, p. 1063604.
- [147] T Eshel. “Mobile Radar Optimized to Detect UAVs, Precision Guided Weapons”. In: *Defense Update* (2013).
- [148] Fraunhofer Institute. *Detection of Small Drones with Millimeter-Wave Radar*. Accessed: 18.06.2021. 2019. URL: <https://www.fhr.fraunhofer.de/en/businessunits/security/Detection-of-small-drones-with-millimeter-wave-radar.html>.
- [149] P. Molchanov, K. Egiazarian, J. Astola, R. I. A. Harmanny, and J. J. M. de Wit. “Classification of small UAVs and birds by micro-Doppler signatures”. In: *2013 European Radar Conference*. 2013, pp. 172–175.
- [150] Matthew Peacock and Michael N Johnstone. “Towards detection and control of civilian unmanned aerial vehicles”. In: *14th Australian Information Warfare Conference*. SRI Security Research Institute. 2013.
- [151] Phuc Nguyen, Hoang Truong, Mahesh Ravindranathan, Anh Nguyen, Richard Han, and Tam Vu. “Matthan: Drone presence detection by identifying physical signatures in the drone’s rf communication”. In: *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. 2017, pp. 211–224.
- [152] Ben Nassi, Raz Ben-Netanel, Adi Shamir, and Yuval Elovici. “Drones’ Cryptanalysis - Smashing Cryptography with a Flicker”. In: *2019 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 1397–1414.
- [153] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. “Flying objects detection from a single moving camera”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015.
- [154] Stephan Sand, Armin Dammann, and Christian Mensing. *Positioning in Wireless Communications Systems*. John Wiley & Sons, 2014.
- [155] Michigan Tech. *Robotic Falconry*. Accessed: 10.07.2016. 2016. URL: <http://me.sites.mtu.edu/rastgaar/home/news/>.
- [156] Guard From Above. *Intercepting hostile drones*. Accessed: 18.06.2021. 2021. URL: <http://guardfromabove.com/>.
- [157] Samy Kamkar. *SkyJack*. Accessed: 18.06.2021. 2013. URL: <http://samy.pl/skyjack/>.
- [158] Daniel P Shepard, Jahshan A Bhatti, Todd E Humphreys, and Aaron A Fansler. “Evaluation of smart grid and civilian UAV vulnerability to GPS spoofing attacks”. In: *Proceedings of the ION GNSS Meeting*. Vol. 3. 2012.
- [159] Sait Murat Giray. “Anatomy of unmanned aerial vehicle hijacking with signal spoofing”. In: *Recent Advances in Space Technologies (RAST), 2013 6th International Conference on*. IEEE. 2013, pp. 795–800.

- [160] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. “Practical trigger-action programming in the smart home”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2014, pp. 803–812.
- [161] SmartThings Inc. *Samsung SmartThings*. Accessed: 18.06.2021. 2020. URL: <https://www.smarthings.com/>.
- [162] Eyal Ronen and Adi Shamir. “Extended Functionality Attacks on IoT Devices: The Case of Smart Lights”. In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 3–12.
- [163] Yang Zhang, Nirvana Meratnia, and Paul JM Havinga. “Outlier detection techniques for wireless sensor networks: A survey.” In: *IEEE Communications Surveys and Tutorials* 12.2 (2010), pp. 159–170.
- [164] Chinh T Vu, Raheem A Beyah, and Yingshu Li. “Composite event detection in wireless sensor networks”. In: *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE International*. IEEE. 2007, pp. 264–271.
- [165] Geoffrey Werner-Allen, Konrad Lorincz, Mario Ruiz, Omar Marcillo, Jeff Johnson, Jonathan Lees, and Matt Welsh. “Deploying a wireless sensor network on an active volcano”. In: *IEEE internet computing* 10.2 (2006), pp. 18–25.
- [166] AVU Phani Kumar, Adi Mallikarjuna Reddy V, and D Janakiram. “Distributed collaboration for event detection in wireless sensor networks”. In: *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. ACM. 2005, pp. 1–8.
- [167] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. “Machine learning in wireless sensor networks: Algorithms, strategies, and applications”. In: *IEEE Communications Surveys & Tutorials* 16.4 (), pp. 1996–2018.
- [168] Liyang Yu, Neng Wang, and Xiaoqiao Meng. “Real-time forest fire detection with wireless sensor networks”. In: *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*. Vol. 2. IEEE. 2005, pp. 1214–1217.
- [169] Georg Wittenburg, Norman Dziengel, Christian Wartenburger, and Jochen Schiller. “A system for distributed event detection in wireless sensor networks”. In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM. 2010, pp. 94–104.
- [170] Majid Bahrepour, Nirvana Meratnia, and Paul JM Havinga. “Sensor fusion-based event detection in Wireless Sensor Networks.” In: *MobiQuitous*. 2009, pp. 1–8.
- [171] National Institute of Standards and Technology (NIST). *Home Smoke Alarm Tests*. Accessed: 18.06.2021. 2013. URL: <https://www.nist.gov/el/home-smoke-alarm-tests>.
- [172] Majid Bahrepour, Nirvana Meratnia, Mannes Poel, Zahra Taghikhaki, and Paul JM Havinga. “Distributed event detection in wireless sensor networks for disaster management”. In: *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*. IEEE. 2010, pp. 507–512.

- [173] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson. “Activity recognition in the home using simple and ubiquitous sensors”. In: *International conference on pervasive computing*. Springer. 2004, pp. 158–175.
- [174] Chao Chen, Barnan Das, and Diane J Cook. “A data mining framework for activity recognition in smart environments”. In: *Intelligent Environments (IE), 2010 Sixth International Conference on*. IEEE. 2010, pp. 80–83.
- [175] A. Fleury, M. Vacher, and N. Noury. “SVM-Based Multimodal Classification of Activities of Daily Living in Health Smart Homes: Sensors, Algorithms, and First Experimental Results”. In: *IEEE Transactions on Information Technology in Biomedicine* 14.2 (2010), pp. 274–283.
- [176] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. “Accurate Activity Recognition in a Home Setting”. In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. UbiComp ’08. New York, NY, USA: ACM, 2008, pp. 1–9.
- [177] Narayanan C Krishnan and Diane J Cook. “Activity recognition on streaming sensor data”. In: *Pervasive and mobile computing* 10 (2014), pp. 138–154.
- [178] James Fogarty, Carolyn Au, and Scott E Hudson. “Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition”. In: *Proceedings of the 19th annual ACM symposium on User interface software and technology*. ACM. 2006, pp. 91–100.
- [179] Younghun Kim, Thomas Schmid, Zainul M Charbiwala, and Mani B Srivastava. “ViridiScope: design and implementation of a fine grained power monitoring system for homes”. In: *Proceedings of the 11th international conference on Ubiquitous computing*. ACM. 2009, pp. 245–254.
- [180] Sidhant Gupta, Matthew S Reynolds, and Shwetak N Patel. “ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home”. In: *Proceedings of the 12th ACM international conference on Ubiquitous computing*. ACM. 2010, pp. 139–148.
- [181] Alex Beltran, Varick L. Erickson, and Alberto E. Cerpa. “ThermoSense: Occupancy Thermal Based Sensing for HVAC Control”. In: *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*. BuildSys’13. New York, NY, USA: ACM, 2013, 11:1–11:8.
- [182] Gabe Cohn, Sidhant Gupta, Jon Froehlich, Eric Larson, and Shwetak N Patel. “GasSense: Appliance-level, single-point sensing of gas activity in the home”. In: *International Conference on Pervasive Computing*. Springer. 2010, pp. 265–282.
- [183] Jon E. Froehlich, Eric Larson, Tim Campbell, Conor Haggerty, James Fogarty, and Shwetak N. Patel. “HydroSense: Infrastructure-mediated Single-point Sensing of Whole-home Water Activity”. In: *Proceedings of the 11th International Conference on Ubiquitous Computing*. UbiComp ’09. New York, NY, USA: ACM, 2009, pp. 235–244.
- [184] Gierad Laput, Yang Zhang, and Chris Harrison. “Synthetic Sensors: Towards General-Purpose Sensing”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2017, pp. 3986–3999.

- [185] Stefan Axelsson. “The Base-Rate Fallacy and the Difficulty of Intrusion Detection”. In: *ACM Trans. Inf. Syst. Secur.* 3.3 (Aug. 2000), 186–205. URL: <https://doi.org/10.1145/357830.357849>.
- [186] BI Intelligence. *Drone market outlook in 2021: industry growth trends, market stats and forecast*. Accessed: 18.06.2021. 2021. URL: <https://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts>.
- [187] Federal Aviation Administration. *FAA Drone Registry Tops One Million*. Accessed: 18.06.2021. 2018. URL: <https://www.transportation.gov/briefing-room/faa-drone-registry-tops-one-million>.
- [188] Financial Times. *China’s DJI targets agriculture as consumer drone sales slow*. Accessed: 18.06.2021. 2019. URL: <https://www.ft.com/content/afa5e042-4c50-11e9-bbc9-6917dce3dc62>.
- [189] BBC. *Japan radioactive drone: Tokyo police arrest man*. Accessed: 18.06.2021. 2015. URL: <https://www.bbc.co.uk/news/world-asia-32465624>.
- [190] Der Spiegel. *Merkel Buzzed by Mini-Drone at Campaign Event*. Accessed: 18.06.2021. 2013. URL: <https://www.spiegel.de/international/germany/merkel-campaign-event-visited-by-mini-drone-a-922495.html>.
- [191] User ‘GWA88’ et al. *Gatwick Airport drone incident*. Accessed: 18.06.2021. 2018. URL: https://en.wikipedia.org/wiki/Gatwick_Airport_drone_incident.
- [192] BBC. *Gang who flew drones carrying drugs into prisons jailed*. Accessed: 18.06.2021. 2018. URL: <https://www.bbc.co.uk/news/uk-england-45980560>.
- [193] Linzi Sheldon. *Woman terrified by drone outside her window*. KIRO 7. Accessed: 18.06.2021. 2014. URL: <http://www.kiro7.com/news/woman-terrified-drone-outside-her-window/81721261>.
- [194] WDRB. *Hillview man arrested for shooting down drone; cites right to privacy*. Accessed: 29.06.2016. 2015. URL: <http://www.wdrb.com/story/29650818/hillview-man-arrested-for-shooting-down-drone-cites-right-to-privacy>.
- [195] Skylogic Research. *DJI market share: here’s exactly how rapidly it has grown in just a few years*. Accessed: 18.06.2021. 2018. URL: <http://thedronegirl.com/2018/09/18/dji-market-share/>.
- [196] Civil Aviation Authority. *The Air Navigation Order 2016 And Regulations (CAP393)*. Accessed: 18.06.2021. 2021. URL: <http://www.caa.co.uk/CAP2038A00>.
- [197] Zhuoling Xiao, Hongkai Wen, Andrew Markham, Niki Trigoni, Phil Blunsom, and Jeff Frolik. “Identification and mitigation of non-line-of-sight conditions using received signal strength”. In: *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. 2013, pp. 667–674.
- [198] Kavitha Muthukrishnan, Berend Jan van der Zwaag, and Paul Havinga. “Inferring motion and location using WLAN RSSI”. In: *Mobile Entity Localization and Tracking in GPS-less Environments*. Springer, 2009, pp. 163–182.

- [199] Vijay Garg. *Wireless communications & networking*. Morgan Kaufmann, 2010.
- [200] Nitis Mukhopadhyay. *Probability and statistical inference*. CRC Press, 2000.
- [201] Kaspersky Lab ICS Cert. *Somebody’s watching! When cameras are more than just ‘smart’*. Accessed: 18.06.2021. 2018. URL: <https://ics-cert.kaspersky.com/reports/2018/03/12/somebodys-watching-when-cameras-are-more-than-just-smart/>.
- [202] Apple Inc. *Apple HomeKit: Your home at your command*. Accessed: 18.06.2021. 2021. URL: <https://www.apple.com/uk/ios/home/>.
- [203] openHAB Foundation e.V. *openHAB: empowering the smart home*. Accessed: 18.06.2021. 2021. URL: <https://www.openhab.org/>.
- [204] Home Assistant. *Awaken your home*. Accessed: 18.06.2021. 2021. URL: <https://www.home-assistant.io/>.
- [205] Silicon Laboratories Inc. *AN1138: Zigbee Mesh Network Performance*. Accessed: 18.06.2021. 2012. URL: <https://www.silabs.com/documents/login/application-notes/an1138-zigbee-mesh-network-performance.pdf>.
- [206] Brian C. Ross. “Mutual Information between Discrete and Continuous Data Sets”. In: *PLOS ONE* 9.2 (2014), pp. 1–5.
- [207] Pramod K. Varshney. *Distributed Detection and Data Fusion*. Springer Science+Business Media New York, 2012.
- [208] A. S. Rawat, P. Anand, H. Chen, and P. K. Varshney. “Collaborative Spectrum Sensing in the Presence of Byzantine Attacks in Cognitive Radio Networks”. In: *IEEE Transactions on Signal Processing* 59.2 (2011), pp. 774–786.
- [209] F. Goksel, M. Ozmen, M. Reeves, B. Shivakumar, and Z. Celik. “On the Safety Implications of Misordered Events and Commands in IoT Systems”. In: *2021 IEEE Security and Privacy Workshops (SPW)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2021, pp. 235–241. URL: <https://doi.ieeecomputersociety.org/10.1109/SPW53761.2021.00038>.
- [210] Shashi Phoha, Nurali Virani, Pritthi Chattopadhyay, Soumalya Sarkar, Brian Smith, and Asok Ray. “Context-aware Dynamic Data-driven Pattern Classification”. In: *Procedia Computer Science* 29 (2014). 2014 International Conference on Computational Science (ICCS), pp. 1324–1333.
- [211] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. “DAG-based attack and defense modeling: Don’t miss the forest for the attack trees”. In: *Computer Science Review* 13-14 (2014), pp. 1–38.
- [212] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. “An Attack Graph-Based Probabilistic Security Metric”. In: *Data and Applications Security XXII*. Springer Berlin Heidelberg, 2008, pp. 283–296.
- [213] Alastair Rensburg, Jason Nurse, and Michael Goldsmith. “Attacker-Parametrised Attack Graphs”. In: *Proceedings of the 2016 International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*. 2016, pp. 316–319.