# Haunted House: Physical Smart Home Event Verification in the Presence of Compromised Sensors

SIMON BIRNBACH, University of Oxford, United Kingdom
SIMON EBERZ, University of Oxford, United Kingdom
IVAN MARTINOVIC, University of Oxford, United Kingdom

In this paper, we verify physical events using data from an ensemble of smart home sensors. This approach both protects against event sensor faults and sophisticated attackers.

In order to validate our system's performance, we set up a "smart home" in an office environment. We recognize 22 event types using 48 sensors over the course of two weeks. Using data from the physical sensors, we verify the event stream supplied by the event sensors to detect both masking and spoofing attacks.

We consider three threat models: a zero-effort attacker, an opportunistic attacker and a sensor-compromise attacker who can arbitrarily modify live sensor data. For spoofed events we achieve perfect classification for 9 out of 22 events and achieve a 0% false alarm rate at a detection rate exceeding 99.9% for 15 events. For 11 events the majority of masking attacks can be detected without causing any false alarms. We also show that even a strong opportunistic attacker is inherently limited to spoofing few select events and that doing so involves lengthy waiting periods. Finally, we demonstrate the vulnerability of a single-classifier system to compromised sensor data and introduce a more secure approach based on sensor fusion.

CCS Concepts: • **Security and privacy** → *Intrusion detection systems*; *Distributed systems security*; *Mobile and wireless security*.

Additional Key Words and Phrases: Internet of things; smart home; event verification

**ACM Reference Format:**
Simon Birnbach, Simon Eberz, and Ivan Martinovic. 2021. Haunted House: Physical Smart Home Event Verification in the Presence of Compromised Sensors. *ACM Trans. Internet Things* 37, 4, Article 111 (August 2021), 28 pages. https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

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 over 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 [30].

Authors' addresses: S. Birnbach, S. Eberz, and I. Martinovic, University of Oxford, Department of Computer Science, Wolfson Building, Parks Road, Oxford OX1 3QD, United Kingdom; emails: {simon.birnbach, simon.eberz, ivan.martinovic}@cs.ox.ac.uk.

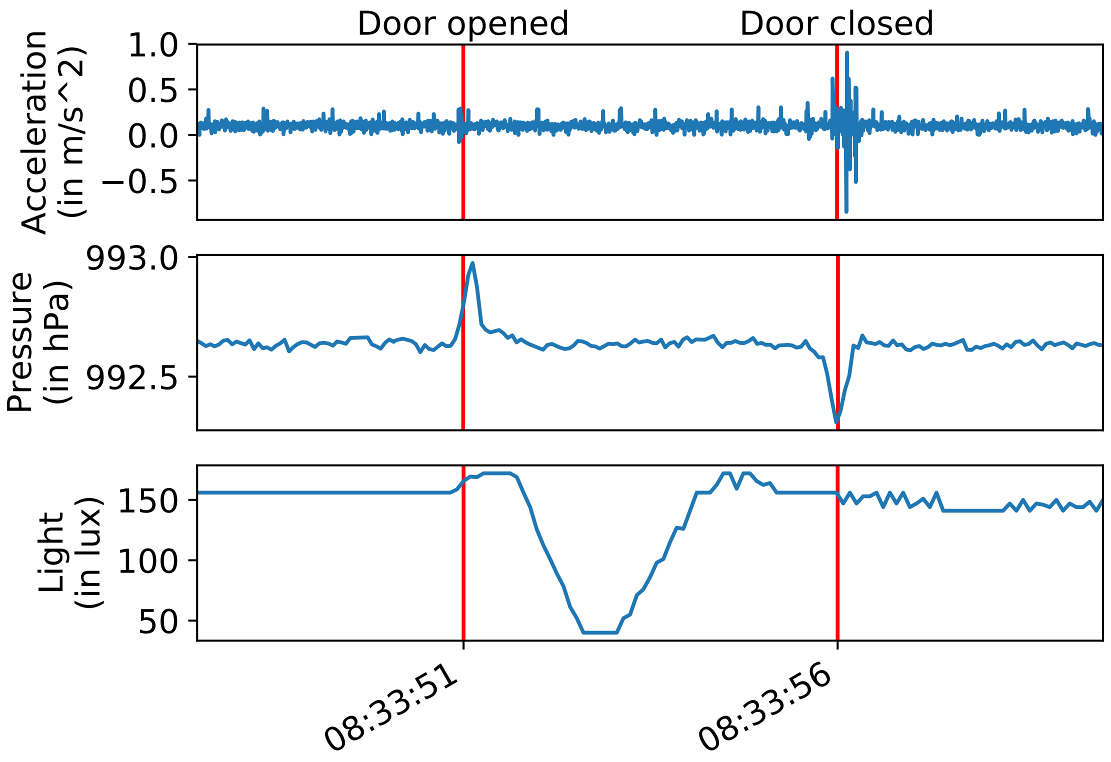


Fig. 1. An example of how door events can affect different sensors. The red lines correspond to the timing of first a door opening event and then a door closing event. One can see that the door closing event has a clear effect on all three sensors, whereas the door opening event affects primarily the light and pressure sensors.

Several approaches have been proposed to reduce the risk posed by insecure IoT devices. Most of these approaches focus on either device and smart application permission systems [15, 28, 40] or network traffic analysis [2, 34, 47]. But none of these approaches can protect event integrity after a device has been compromised. 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 [8]). An overview of smart home devices and potential attacker goals can be found in Table 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 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 1 shows how a closing door affects a pressure sensor, a light sensor and an accelerometer placed within the same room. Table 1 provides a (non-exhaustive) overview of which sensors are expected to sense effects of events relating to current smart home devices.

Table 1. Smart home devices and possible attacker goals. Attacker goals follow the classification by Denning et. al. [10]. 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** | **References** |
|---|---|---|---|
| **Door** | Enabling physical entry, Physical theft | Accelerometer, Light, Air pressure, Microphone | [8] |
| **Window** | Enabling physical entry, Physical theft | Accelerometer, Air quality, Air pressure, Microphone | [11, 28] |
| **Window shade** | Espionage, Voyeurism | Accelerometer, Power, Microphone, Light | [5, 35] |
| **Light** | Data exfiltration, Espionage, Causing physical harm | Light, Power | [42, 43] |
| **Smart cam** | Espionage, Voyeurism | Power, Accelerometer | [9] |
| **Thermostat** | Enabling physical entry, Extortion | Temperature | [10, 28] |
| Door lock | Enabling physical entry, Physical theft | Accelerometer, Microphone | [23] |
| Siren | Causing physical harm, Causing environment damage, Misinformation | Light, Microphone | [14] |
| Kettle, Oven, Iron | Causing physical harm, Causing environment damage | Power, Temperature, Humidity, Accelerometer, Microphone | [8, 11] |

In this work, we introduce Peeves, a system that automatically verifies events in smart homes based on their physical 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 the sensors that are relevant for verification without supervision by the user. We show the feasibility of Peeves in a real-world experiment.

In this paper, we make the following contributions:

- We introduce Peeves, a system to automatically learn smart home event signatures for various sensing modalities.
- 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 to increase their success rate and show that Peeves is still effective against this strong adversary.
- 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.

The remainder of the paper is organized as follows: Section 2 gives an overview of smart home background and related work. Sections 3 and 4 outline our system and threat model. We describe our experimental design in Section 5 and our methods in Section 6. We present our results in Section 7, discuss real-world issues and limitations in Section 8, and conclude the paper in Section 9.

[1]https://www.doi.org/10.5287/bodleian:mv22Jk2Xj

## 2 BACKGROUND

### 2.1 Smart home systems

Smart home systems enable users to remotely control and monitor devices and automate everyday tasks. These devices range from surveillance cameras over smart locks 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, and offload the burden of worrying about energy-efficient heating.

There are many competing smart home systems ranging from commercial solutions such as Samsung's SmartThings [26] and Apple's HomeKit [25] to open-source tools like OpenHAB [37] and HomeAssistant [3].

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.

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.

### 2.2 Related work

*2.2.1 Security of smart home systems.* There has been extensive research focusing on the security of smart home systems and devices. Fernandes et al. [14] investigated the popular SmartThings system and uncovered several vulnerabilities due to overprivileged smart home applications. One of these vulnerabilities allows malicious applications running on the smart hub to spoof events as if they were originating from other devices in the network, thus triggering unintended actions such as erroneously setting off a fire alarm. This risk extends even beyond malicious control software as presented in their paper, as compromised devices can easily send false event updates, or suppress the notification of an actual event. More recently, Goksel et al. [19] have shown how even events that are just delayed or reordered can affect the safety of smart home systems; this vulnerability could be exploited by a determined attacker.

As demonstrated by Sivaraman et al. [49], malicious mobile phone applications can compromise home networks and thus open up smart home systems to attacks from remote attackers over the internet. Ronen et al. [43] showed how physical proximity can be used to compromise smart devices. They exploited communication protocol flaws present in

Philips Hue smart lamps to create a worm that can spread across a city from household to household. In [42], Ronen and Shamir further showed how attackers can misuse compromised smart devices such as smart lights and go beyond their intended functionality.

*2.2.2 Security of trigger-action programming.* Previous work has highlighted the risks posed by unsafe and insecure interactions between devices and smart applications. Device actuations can lead to unintentional consequences if trigger-action rules are being used in different contexts [28], if they include external trigger-action platforms [8], or if they include hidden inter-app interactions through physical channels [11]. These issues have been addressed through more fine-grained permission systems and policy frameworks [28, 50, 54], better logging for forensic purposes [52], and static analysis of smart applications [7, 8, 11, 12, 36] to detect possible physical channel interactions. IoTSafe [12] further combines static analysis with dynamic testing to tailor the discovery of physical side-channels to the current deployment. A recent survey by He et al. [22] gives a good overview on how context sensing can be used for access control in these systems. While these systems help to reduce the risk that physical channel interactions pose, they do not protect against cases where an existing and desired trigger-action rule is being exploited, e.g., by a compromised smart home device.

*2.2.3 Device and behavior fingerprinting.* There has been an increased interest in identifying smart home devices and fingerprinting of device behavior. Several solutions have been proposed that identify devices based on their network traffic fingerprint [4, 34, 47]. Other work has focused on inferring smart home activities and device event signatures from network traffic to highlight the privacy implications of wireless communication in smart homes [1, 2]. Aegis [48] monitors smart home usage patterns based on event sequences and detects malicious events by comparing observed behavior to previously learned fingerprints of benign user behavior.

Closest to our work is HoMonit [56], 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.

*2.2.4 Event detection with heterogeneous sensors.* Recent research has exploited the fact that smart home events have physical effects that can be measured through various sensing modalities. Perceptio [21] builds on the idea of context-based pairing [33, 45] which relies on common sensor measurements to provide the entropy needed to derive symmetric keys. But as devices present in a smart home have diverse sensing modalities, pairing protocols must consider this device heterogenenity in order to be usable. 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. These fingerprints are then used to provide the entropy for its pairing protocol.

In a recent paper Fu et al. presented HAWatcher [17], a smart home anomaly detection system that uses event correlation combined with semantic analysis of smart applications to generate rule sets that can be checked at runtime to detect unusual system behavior. This system also takes into account the state of sensors that are used in trigger-action rules by creating corresponding threshold-based events. While this allows HAWatcher to consider event-sensor correlations in a limited and coarse-grained fashion, the system is not able to detect correlations with sensors that are not already part of a trigger-action rule. This means that the majority of sensors are not leveraged for anomaly detection, which leaves the system more vulnerable to compromised sensors.

In a non-security context the observation that physical events can affect a diverse range of sensors in home environments has been used for activity recognition systems. This has been used to detect both human and device activities in homes through diverse means such as microphones [16], electromagnetic interference [20], as well as many other ambient sensor modalities [29, 31]. A recent paper aiming to provide general-purpose sensing in home environments is SyntheticSensors [32]. In this paper, synthetic sensors are introduced as an abstraction to physical events 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 is 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". This paper gives an interesting outlook on how and what kind of events can be detected in a smart home setting.

However, unlike Peeves, none of the aforementioned activity recognition systems are designed with an adversary in mind and events have to be labeled explicitly by the user. In contrast, Peeves can verify the veracity of reported smart home events to defend against event spoofing and masking attacks and uses smart home event notifications to learn event signatures automatically.

*2.2.5 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 [38, 55] and physical anomaly detection techniques [18, 39] 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. [39] 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 [53]. Yet, there have already been studies showing how these detection systems can be evaded by a determined attacker [13].

There is very little research on the effect of compromised nodes on event detection in WSNs. In [24], 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.

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. [46] 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 their 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 SYSTEM MODEL

We consider a system model inspired by contemporary smart home systems as described in Section 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., ~20-200ms for Zigbee with a maximal payload size of 150 bytes and a maximal hop size of 6 [27]). 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 [14] 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 2.

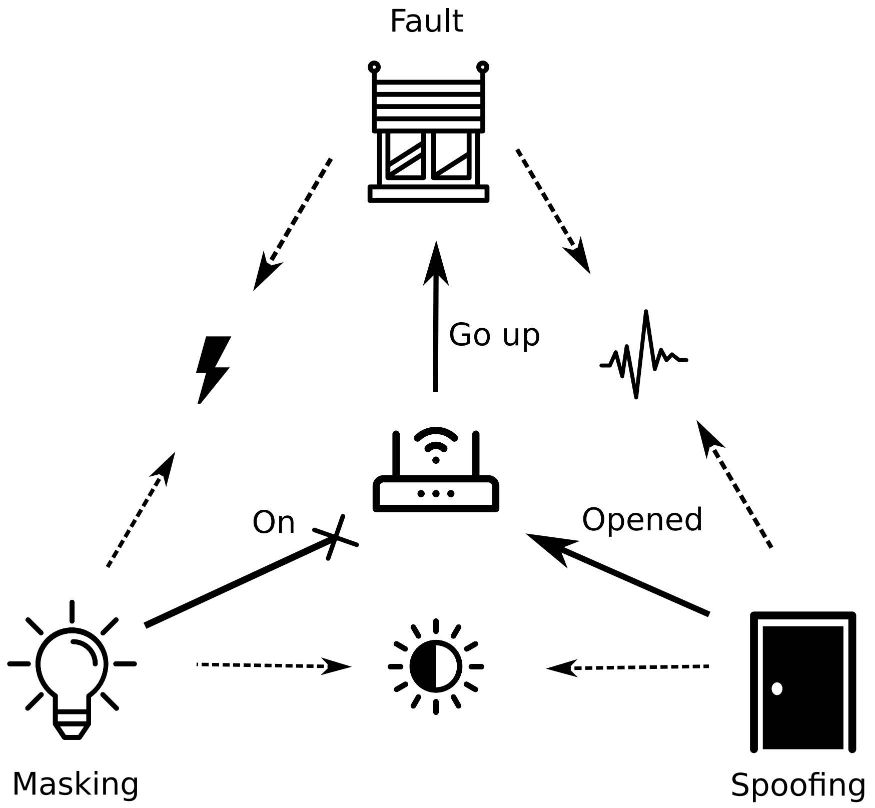


Fig. 2. This figure gives an overview over Peeves and three exemplary scenarios. The window shade at the top exhibits a fault and does not act on a command by the smart hub in the middle. On the bottom right, the door sensor sends a spoofed event to the hub, falsely claiming that the door has been opened. 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).

## 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 [8], 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.

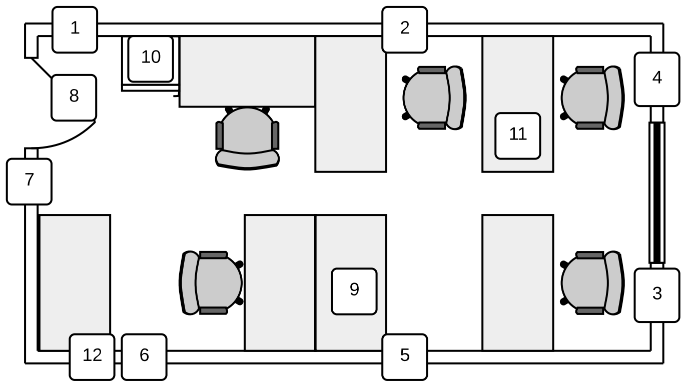


Fig. 3. The device deployment for the experiment is pictured in this floorplan. The numbers refer to the group numbers in Tables 3 and 4. Sensor group 8 is mounted on the door, groups 9 and 11 are placed on tables, and group number 4 is fixed to a shelf. All the other sensors are mounted on the wall.

Table 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) |

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.

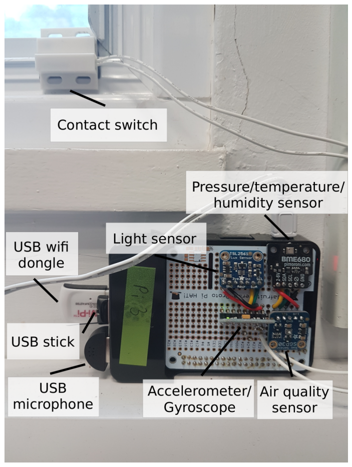
Our system considers three types of attackers:

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

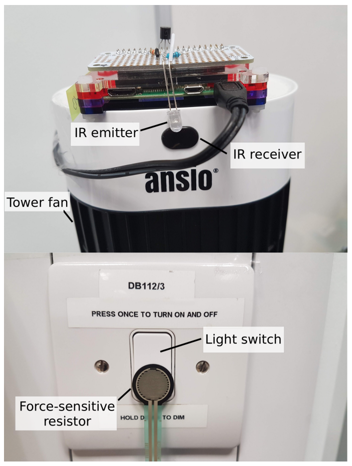
For all attacker types, event sensors are considered to be untrusted, as they can be compromised or exhibit random faults. None of the attackers has physical access to the devices. The zero-effort attacker spoofs or masks events at arbitrary times and does not have knowledge of any system parameters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

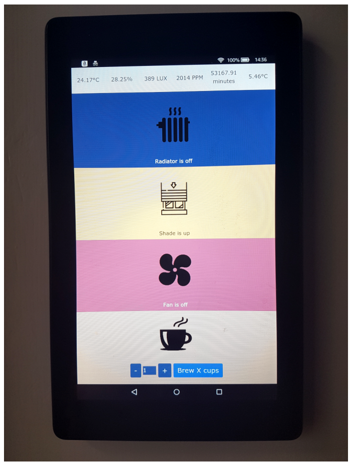
Finally, a sensor-compromise attacker is an extension of the opportunistic attacker. In addition to the above capabilities, a sensor-compromise attacker can also modify the sensor data of a subset of verification sensors in real-time (e.g., replay a correct sensor signature of a compromised accelerometer for the “door closed” event). The attacker uses this to create



(a) The sensor configuration for one of the Raspberry Pis. A detailed listing of the sensor configuration can be found in Table 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 Rasberry 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.

Fig. 4. Experimental setup

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 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.

### 5.1 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 3, while a detailed sensor configuration is listed in Table 3. A description of the sensor types used in the experiment can be found in Table 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 4.

### 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 4(a). Light switch events are registered through a force-sensitive resistor that registers when

Table 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 2. The deployment is pictured in Figure 3.

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

Table 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 3.

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

Table 5. This table gives an overview over which events were triggered manually and which were automatically triggered following a schedule. 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 | ✓ | ✗ | – |

the switch is being pressed, see Figure 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 it from recording video from the room. Window shade, radiator, fan and coffee machine can be controlled via a tablet, see Figure 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 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 and to ensure that events are captured in a wide range of contexts. More details on event schedules are given in Table 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 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.

### 5.3 Ethical concerns

This project has been reviewed by and received clearance by the responsible research ethics committee at our university. 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.

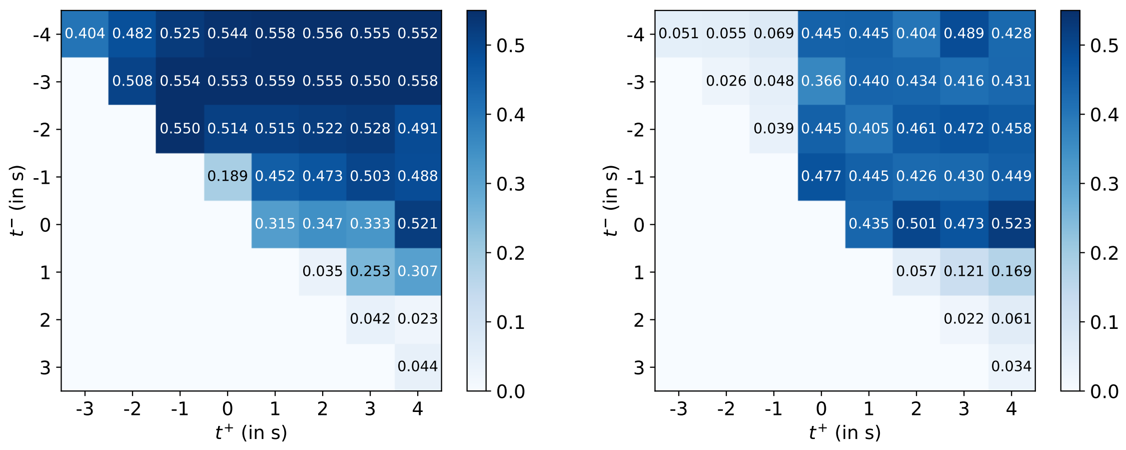
## 6 METHODS

### 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 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.

### 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



(a) Light sensor (b) Accelerometer

Fig. 5. Window parameter grid search for the "Door closed" event. Values given are the average RMI.

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.

## 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 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. 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.

### 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 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

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

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 [44].

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%.

### 6.5 Classification and metrics

In this work, we compare two decision making processes. The first approach uses a monolithic classifier as presented in our original conference paper [6]. 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).

*6.5.1 Monolithic classifier.* For the monolithic classifier approach, we train a single SVM classifier on all the features extracted through the process described in Section 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 7) to obtain the EER.

*6.5.2 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 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 [51]. We choose this fusion rule on account of its recognized strong performance and ease of use [41]. 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.

## 6.6 Attacker implementation

As outlined in our threat model (Section 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.

*6.6.1 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. 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 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.

*6.6.2 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.

## 7 RESULTS

### 7.1 Feature distinctiveness

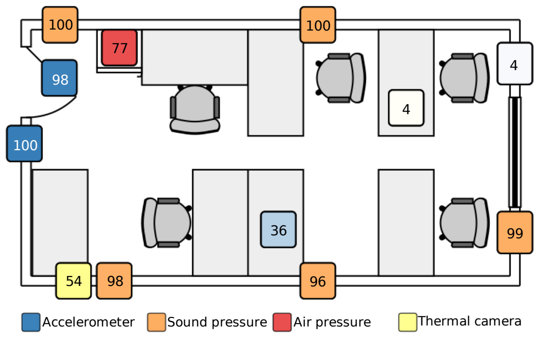
Table 6 shows the distinctiveness of feature groups for each event. Each cell shows the maximum RMI within the sensor group. Most results are relatively intuitive: Accelerometers detect door events, window shade events and the use of the coffee machine. Power meters primarily detect the devices they are connected to. Light sensors detect both toggling of the light switch and events that lead to obscuring the 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

Table 6. Maximum RMI for each event within the different sensor groups. Sensor group labels are given in Table 2.

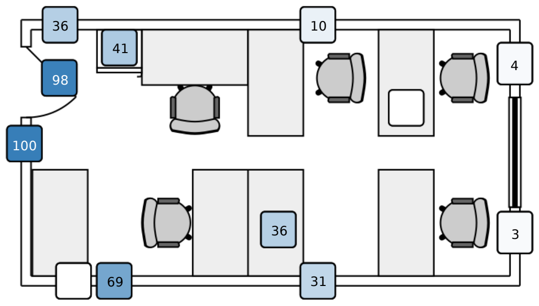
| Event | A | G | M | P | H | T | L | PM | AQ | S | R | TC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Coffee machine used | 87 | 67 | 4 | 12 | 11 | 24 | 82 | 100 | 75 | 37 | 9 | 37 |
| Doorbell used | 43 | 50 | 45 | 30 | 25 | 33 | 65 | 37 | 70 | 38 | 74 | 24 |
| Door closed | 99 | 96 | 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. 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.



(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 %.

Fig. 6. Type and quality of most distinctive sensor depending on its position

Figure 6(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 6(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.

## 7.2 Spoofing attack detection

The classification results for the spoofing attack detection are shown in Table 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, but extremely reliable features. In order to more closely investigate events with less ideal performance (such as window events), we show their ROC curve in Figure 7. 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.

## 7.3 Masking attack detection

The classification results for the masking attack detection are shown in Table 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

Table 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) |

Table 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 |

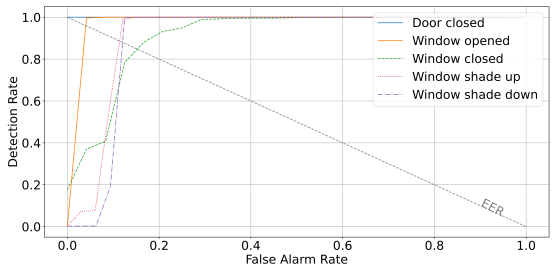


Fig. 7. Receiver Operating Characteristics (ROC) curve for the spoofing attack detection of different events. Here we only show events with an EER significantly higher than 0 (well-classified events with an EER of 0 are a flat line at the top).

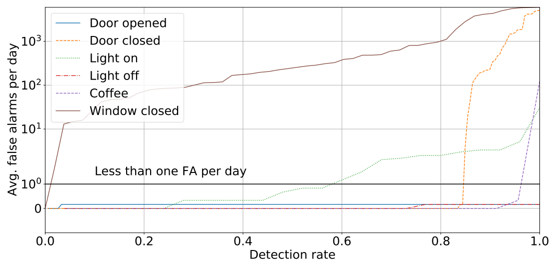


Fig. 8. Average false alarm counts per day at different detection rates for masking attacks targeting various events. The black horizontal line shows when false alarms reach an average of one false alarm per day over the testing period.

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 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

Table 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 |

Table 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.

| | **th=0.5%** | | **th=1%** | | **th=5%** | |
|---|---|---|---|---|---|---|
| Event | 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 |

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 8 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.

## 7.4 Opportunistic attacker

The goal of the attacker's surrogate model (see Section 6.6.1) 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. Due to space constraints, we focus on spoofing attacks in this section.

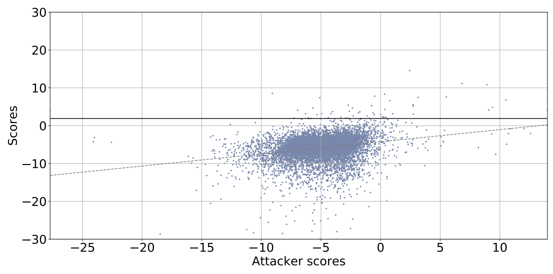


Fig. 9. Score correlation between the legitimate model and the opportunistic attacker's surrogate model. The dashed grey line is the linear regression line and the solid black line shows the 99% detection rate cut-off.

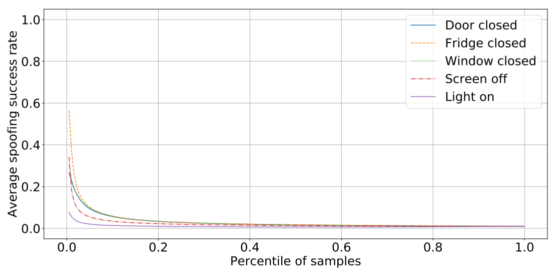


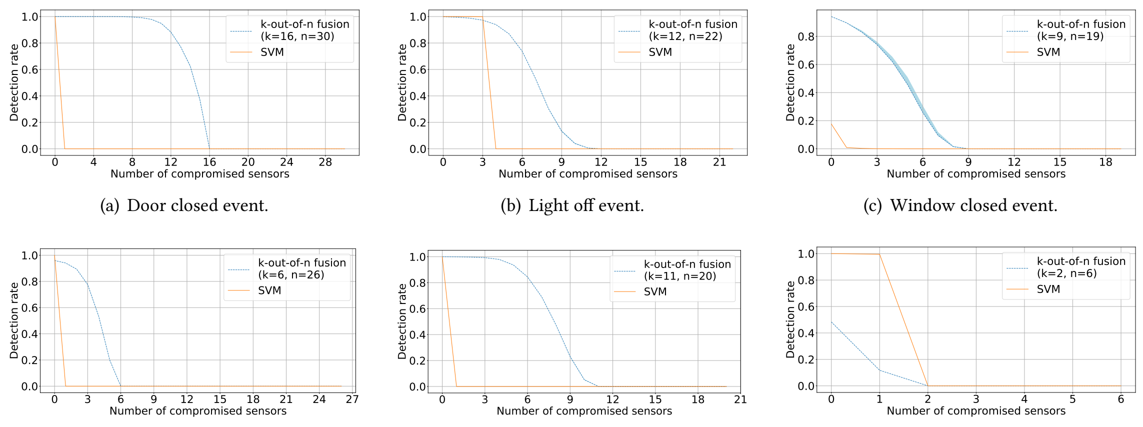
Fig. 10. Success rates of the opportunistic attacker for various thresholds. A higher thresholds leads to higher attack success rate but increased wait time for an opportunity.

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 9. 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 9. Door, fridge and window and screen events show a particularly high correlation. For these events, we show in Figure 10 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 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 10 and Figure 10 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.

## 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 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.



(a) Door closed event.

(b) Light off event.

(c) Window closed event.

(d) Door opened event.

(e) Light on event.

(f) PC on event.

Fig. 11. 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 $C$. The SVM threshold and the value of k are chosen as the largest value that results in 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 a shaded area. More details can be found in the text.

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, 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 6.6.2. While we only present results for spoofed events due to space constraints, the attack equally applies to masking. 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 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 $C_\alpha$ at which the attacker is able to reduce the detection rate to zero and thus blind the system.

In Figure 11, 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 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

Table 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.

| | **C=0** | | **C=1** | | **C=2** | | **C=5** | | **C=10** | | **C=15** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Event | 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 |

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 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 13 out of 22 events with regards to the minimal $C_\alpha$ that the attacker needs to blind the system. Adopting a single SVM classifier only leads to better values for $C_\alpha$ in two cases (doorbell and radiator off), both of which already exhibit very bad performance without an attacker present, and regardless whether a single classifier or sensor fusion is being used. 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 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.

## 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 work, 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.

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, then 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. [19] 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.

## 9 CONCLUSION

In this paper 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. 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.

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

## ACKNOWLEDGMENTS

This work was supported by a grant from Mastercard and the Engineering and Physical Sciences Research Council.

## REFERENCES

[1] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. 2020. 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* (Linz, Austria) *(WiSec '20)*. ACM, New York, NY, USA, 207–218. https://doi.org/10.1145/3395351.3399421

[2] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. (2017). arXiv:1708.05044 [cs.CR] http://arxiv.org/abs/1708.05044

[3] Home Assistant. 2021. *Awaken your home.* Retrieved January 14, 2021 from https://www.home-assistant.io/

[4] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. Behavioral Fingerprinting of IoT Devices. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security* (Toronto, Canada) *(ASHES '18)*. ACM, New York, NY, USA, 41–50. https://doi.org/10.1145/3266444.3266452

[5] Simon Birnbach, Richard Baker, and Ivan Martinovic. 2017. Wi-Fly?: Detecting Privacy Invasion Attacks by Consumer Drones. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium (NDSS)* (San Diego, California, USA). The Internet Society. https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/wi-fly-detecting-privacy-invasion-attacks-consumer-drones/

[6] Simon Birnbach, Simon Eberz, and Ivan Martinovic. 2019. Peeves: Physical Event Verification in Smart Homes. In *Proceedings of the 2019 ACM Conference on Computer and Communications Security (CCS),*. ACM, New York, NY, USA.

[7] Z. Berkay Celik, Patrick McDaniel, and Gang Tan. 2018. Soteria: Automated IoT Safety and Security Analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 147–158. https://www.usenix.org/conference/atc18/presentation/celik

[8] Z. Berkay Celik, Gang Tan, and Patrick D. McDaniel. 2019. 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. https://www.ndss-symposium.org/ndss-paper/iotguard-dynamic-enforcement-of-security-and-safety-policy-in-commodity-iot/

[9] Kaspersky Lab ICS Cert. 2018. *Somebody's watching! When cameras are more than just 'smart'*. Retrieved January 14, 2021 from https://ics-cert.kaspersky.com/reports/2018/03/12/somebodys-watching-when-cameras-are-more-than-just-smart/

[10] Tamara Denning, Tadayoshi Kohno, and Henry M. Levy. 2013. Computer Security and the Modern Home. *Commun. ACM* 56, 1 (Jan. 2013), 94–103. https://doi.org/10.1145/2398356.2398377

[11] Wenbo Ding and Hongxin Hu. 2018. On the Safety of IoT Device Physical Interaction Control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. ACM, New York, NY, USA, 832–846. https://doi.org/10.1145/3243734.3243865

[12] Wenbo Ding, Hongxin Hu, and Long Cheng. 2021. IOTSAFE: 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. https://www.ndss-symposium.org/ndss-paper/iotsafe-enforcing-safety-and-security-policy-with-real-iot-physical-interaction-discovery/

[13] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. 2020. Constrained Concealment Attacks against Reconstruction-Based Anomaly Detectors in Industrial Control Systems. In *Annual Computer Security Applications Conference* (Austin, USA) *(ACSAC '20)*. Association for Computing Machinery, New York, NY, USA, 480–495. https://doi.org/10.1145/

3427228.3427660

[14] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security Analysis of Emerging Smart Home Applications. In *2016 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 636–654. https://doi.org/10.1109/SP.2016.44

[15] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 531–548. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/fernandes

[16] James Fogarty, Carolyn Au, and Scott E. Hudson. 2006. 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* (Montreux, Switzerland) *(UIST '06)*. Association for Computing Machinery, New York, NY, USA, 91–100. https://doi.org/10.1145/1166253.1166269

[17] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2021. HAWatcher: Semantics-Aware Anomaly Detection for Appified Smart Homes. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 4223–4240. https://www.usenix.org/conference/usenixsecurity21/presentation/fu-chenglong

[18] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. 2018. A Survey of Physics-Based Attack Detection in Cyber-Physical Systems. *ACM Comput. Surv.* 51, 4, Article 76 (July 2018), 36 pages. https://doi.org/10.1145/3203245

[19] F. Goksel, M. Ozmen, M. Reeves, B. Shivakumar, and Z. Celik. 2021. On the Safety Implications of Misordered Events and Commands in IoT Systems. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE Computer Society, Los Alamitos, CA, USA, 235–241. https://doi.org/10.1109/SPW53761.2021.00038

[20] Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. 2010. 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* (Copenhagen, Denmark) *(UbiComp '10)*. Association for Computing Machinery, New York, NY, USA, 139–148. https://doi.org/10.1145/1864349.1864375

[21] Jun Han, Albert Jin Chung, Manal Kumar Sinha, Madhumitha Harishankar, Shijia Pan, Hae Young Noh, Pei Zhang, and Patrick Tague. 2018. 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, 836–852. https://doi.org/10.1109/SP.2018.00041

[22] Weijia He, Valerie Zhao, Olivia Morkved, Sabeeka Siddiqui, Earlence Fernandes, Josiah Hester, and Blase Ur. 2021. SoK: Context Sensing for Access Control in the Adversarial Home IoT. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE.

[23] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. 2016. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (Xi'an, China) *(ASIA CCS '16)*. ACM, New York, NY, USA, 461–472. https://doi.org/10.1145/2897845.2897886

[24] Vittorio P. Illiano, Luis Muñoz-González, and Emil C. Lupu. 2017. Don't fool Me!: Detection, Characterisation and Diagnosis of Spoofed and Masked Events in Wireless Sensor Networks. *IEEE Transactions on Dependable and Secure Computing* 14, 3 (2017), 279–293. https://doi.org/10.1109/TDSC.2016.2614505

[25] Apple Inc. 2021. *Apple HomeKit: Your home at your command.* Retrieved January 14, 2021 from https://www.apple.com/uk/ios/home/

[26] SmartThings Inc. 2020. *Samsung SmartThings.* Retrieved January 14, 2021 from https://www.smartthings.com/

[27] Silicon Laboratories Inc. 2012. *AN1138: Zigbee Mesh Network Performance.* Retrieved January 14, 2021 from https://www.silabs.com/documents/login/application-notes/an1138-zigbee-mesh-network-performance.pdf

[28] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z Morley Mao, and Atul Prakash. 2017. ContexloT: 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. https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/contexlot-towards-providing-contextual-integrity-appified-iot-platforms/

[29] Younghun Kim, Thomas Schmid, Zainul M. Charbiwala, and Mani B. Srivastava. 2009. ViridiScope: Design and Implementation of a Fine Grained Power Monitoring System for Homes. In *Proceedings of the 11th International Conference on Ubiquitous Computing* (Orlando, Florida, USA) *(UbiComp '09)*. Association for Computing Machinery, New York, NY, USA, 245–254. https://doi.org/10.1145/1620545.1620582

[30] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50, 7 (2017), 80–84. https://doi.org/10.1109/MC.2017.201

[31] Narayanan C. Krishnan and Diane J. Cook. 2014. Activity Recognition on Streaming Sensor Data. *Pervasive Mob. Comput.* 10 (Feb. 2014), 138–154. https://doi.org/10.1016/j.pmcj.2012.07.003

[32] Gierad Laput, Yang Zhang, and Chris Harrison. 2017. Synthetic Sensors: Towards General-Purpose Sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 3986–3999. https://doi.org/10.1145/3025453.3025773

[33] Markus Miettinen, N. Asokan, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and Majid Sobhani. 2014. 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* (Scottsdale, Arizona, USA) *(CCS '14)*. ACM, New York, NY, USA, 880–891. https://doi.org/10.1145/2660267.2660334

[34] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2177–2184. https://doi.org/10.1109/ICDCS.2017.283

[35] Ben Nassi, Raz Ben-Netanel, Adi Shamir, and Yuval Elovici. 2019. Drones' Cryptanalysis - Smashing Cryptography with a Flicker. In *2019 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 1397–1414. https://doi.org/10.1109/SP.2019.00051

[36] Dang Tu Nguyen, Chengyu Song, Zhiyun Qian, Srikanth V. Krishnamurthy, Edward J. M. Colbert, and Patrick McDaniel. 2018. IotSan: Fortifying the Safety of IoT Systems. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (Heraklion, Greece) *(CoNEXT '18)*. ACM, New York, NY, USA, 191–203. https://doi.org/10.1145/3281411.3281440

[37] openHAB Foundation e.V. 2021. *openHAB: empowering the smart home*. Retrieved January 14, 2021 from https://www.openhab.org/

[38] Miroslav Pajic, Insup Lee, and George J. Pappas. 2017. Attack-Resilient State Estimation for Noisy Dynamical Systems. *IEEE Transactions on Control of Network Systems* 4, 1 (2017), 82–92. https://doi.org/10.1109/TCNS.2016.2607420

[39] Junkil Park, Radoslav Ivanov, James Weimer, Miroslav Pajic, Sang Hyuk Son, and Insup Lee. 2017. Security of Cyber-Physical Systems in the Presence of Transient Sensor Faults. *ACM Trans. Cyber-Phys. Syst.* 1, 3, Article 15 (May 2017), 23 pages. https://doi.org/10.1145/3064809

[40] Amir Rahmati, Earlence Fernandes, Kevin Eykholt, and Atul Prakash. 2018. Tyche: A Risk-Based Permission Model for Smart Homes. In *2018 IEEE Cybersecurity Development (SecDev)*. IEEE, 29–36. https://doi.org/10.1109/SecDev.2018.00012

[41] A. S. Rawat, P. Anand, H. Chen, and P. K. Varshney. 2011. Collaborative Spectrum Sensing in the Presence of Byzantine Attacks in Cognitive Radio Networks. *IEEE Transactions on Signal Processing* 59, 2 (2011), 774–786. https://doi.org/10.1109/TSP.2010.2091277

[42] Eyal Ronen and Adi Shamir. 2016. Extended Functionality Attacks on IoT Devices: The Case of Smart Lights. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 3–12. https://doi.org/10.1109/EuroSP.2016.13

[43] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In *2017 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 195–212. https://doi.org/10.1109/SP.2017.14

[44] Brian C. Ross. 2014. Mutual Information between Discrete and Continuous Data Sets. *PLOS ONE* 9, 2 (02 2014), 1–5. https://doi.org/10.1371/journal.pone.0087357

[45] D. Schürmann and S. Sigg. 2013. Secure Communication Based on Ambient Audio. *IEEE Transactions on Mobile Computing* 12, 2 (2013), 358–370. https://doi.org/10.1109/TMC.2011.271

[46] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N. Asokan. 2019. Sensor-Based Proximity Detection in the Face of Active Adversaries. *IEEE Transactions on Mobile Computing* 18, 2 (2019), 444–457. https://doi.org/10.1109/TMC.2018.2839604

[47] Sandra Siby, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. 2017. IoTScanner: Detecting Privacy Threats in IoT Neighborhoods. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security* (Abu Dhabi, United Arab Emirates) *(IoTPTS '17)*. ACM, New York, NY, USA, 23–30. https://doi.org/10.1145/3055245.3055253

[48] Amit Kumar Sikder, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. 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)*. Association for Computing Machinery, New York, NY, USA, 28–41. https://doi.org/10.1145/3359789.3359840

[49] Vijay Sivaraman, Dominic Chan, Dylan Earl, and Roksana Boreli. 2016. Smart-Phones Attacking Smart-Homes. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (Darmstadt, Germany) *(WiSec '16)*. ACM, New York, NY, USA, 195–200. https://doi.org/10.1145/2939918.2939925

[50] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. 2017. SmartAuth: User-Centered Authorization for the Internet of Things. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 361–378. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tian

[51] Pramod K. Varshney. 2012. *Distributed Detection and Data Fusion*. Springer Science+Business Media New York. https://doi.org/10.1007/978-1-4612-1904-0

[52] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl A. Gunter. 2018. 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. https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_01A-2_Wang_paper.pdf

[53] Chathurika S. Wickramasinghe, Daniel L. Marino, Kasun Amarasinghe, and Milos Manic. 2018. Generalization of Deep Learning for Cyber-Physical System Security: A Survey. In *44th Annual Conference of the IEEE Industrial Electronics Society (IECON)*. 745–751. https://doi.org/10.1109/IECON.2018.8591773

[54] Yinhao Xiao, Yizhen Jia, Chunchi Liu, Arwa Alrawais, Molka Rekik, and Zhiguang Shan. 2020. HomeShield: A Credential-Less Authentication Framework for Smart Home Systems. *IEEE Internet of Things Journal* 7, 9 (2020), 7903–7918. https://doi.org/10.1109/JIOT.2020.3003621

[55] Sze Zheng Yong, Minghui Zhu, and Emilio Frazzoli. 2018. Switching and Data Injection Attacks on Stochastic Cyber-Physical Systems: Modeling, Resilient Estimation, and Attack Mitigation. *ACM Trans. Cyber-Phys. Syst.* 2, 2, Article 9 (June 2018), 2 pages. https://doi.org/10.1145/3204439

[56] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. 2018. HoMonit: Monitoring Smart Home Apps from Encrypted Traffic. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. ACM, New York, NY, USA, 1074–1088. https://doi.org/10.1145/3243734.3243820