Activity Recognition Protection for IoT Trigger-Action Platforms

Mahmoud Aghvamipanah*, Morteza Amini*, Cyrille Artho†, Musard Balliu†
*Sharif University of Technology, Tehran, Iran
Email: m.aghvami@sharif.edu, amini@sharif.edu
†KTH Royal Institute of Technology, Stockholm, Sweden
Email: artho@kth.se, musard@kth.se

***Abstract*—Smart home devices collect and transmit user data to smart home Trigger Action Platforms (TAPs) for processing and executing automation rules. However, this data can also be used to infer user activities or other sensitive information. In this paper, we propose PTAP, a privacy-preserving approach based on adversarial example attacks. PTAP injects targeted perturbations into time-series sensor data, effectively confounding potentially malicious TAP classifiers. Our approach significantly reduces the chance of user activity recognition for a malicious TAP while preserving the essential information for automation rule execution, thus safeguarding TAP utility. We evaluated PTAP using a real-world smart-home dataset and examined its effectiveness in preserving utility through the execution of various IoT applications. Our results demonstrate that PTAP effectively preserves user privacy (reducing the accuracy of a malicious classifier 91 to 6 percent) while maintaining automation rule integrity, providing a practical and effective solution to protect user privacy in smart-home environments.**

I. Introduction

Smart home Trigger Action Platforms (TAPs), such as Samsung SmartThings and Apple HomeKit, and more general platforms like IFTTT, are becoming increasingly popular among smart home users, boosting over 280 million users globally [27]. These platforms allow end users to create and execute customized automation rules, also called IoT applications, in third-party cloud environments, thus providing a seamless and user-friendly experience for managing smart homes.

IoT applications are reactive applications that execute code based on triggers and actions. For example, an IoT application designed to manage users' power consumption, such as "When the power rises above the specified threshold, turn off the appliance(s) connected to the selected plug" [54], provides the TAP platform with continuous access to the trigger pertaining to a user's power consumption, and executes the action of turning off appliances. While this IoT application offers clear utility to the user by assisting in managing power consumption, it also poses privacy risks. Seemingly nonsensitive information such as power consumption enables a malicious TAP to collect the data provided by this specific application and to infer the user's daily activities, thus breaching the user's privacy.

Unfortunately, these privacy risks are real. Recently, Samsung SmartThings announced the SmartThings Context API [27, 56] which leverages machine learning classifiers on smart home data to offer inferences as new services for users. Notably, they emphasized the platform's potential in using classifiers for home occupancy status and user contexts, including user activity recognition. While SmartThings introduces inferences as a user feature, it can be viewed as a significant privacy concern, given the potential misuse of such inferred information by compromised or malicious TAPs. In fact, a survey [26] with 40 smart-home end users shows that 47 % of them are concerned about household profiling, and 42 % are concerned about the sale of their data. Moreover, major appliance manufacturers such as LG and Whirlpool have found that fewer than half of their smart appliance owners have connected their devices to the Internet, predominantly due to privacy concerns [48].

Recent research has focused on different threat models that target user privacy in TAPs [5], including malicious app makers [6, 11], network eavesdroppers [4] [3], physical environment attackers [31, 65], and untrusted TAPs [14, 15, 16, 51, 53, 68]. The latter threat model, which assumes the TAPs are untrustworthy, is the most powerful and realistic one because TAPs may share their data with third parties [55] or get compromised [20]. A malicious or compromised TAP with access to seemingly nonsensitive information, e.g., power consumption, can infer sensitive user-specific information including occupancy status and user identities [42, 66], vacation habits [67], and household routines [8, 38, 63].

Figure 1 provides an overview of the TAP utility and the associated privacy concerns in a smart home. The user sends seemingly nonsensitive sensor data to the TAP to execute IoT applications for utility. However, a compromised or untrusted TAP can aggregate this data and extract sensitive inferences using machine learning classifiers, posing a significant privacy concern.

In this paper, we introduce the Privacy-preserving Trigger-action Platform (PTAP) as a solution to address the potential threat of untrusted TAPs in smart home environments. Unlike previous studies, which focused on protecting sensitive data points, PTAP considers the privacy of seemingly nonsensitive aggregated data sets over time. As in our example, this data can reveal personal behavior patterns and routines by means of sensitive inferences such as home occupancy, daily activities, and economic status. Specifically, we illustrate the PTAP solution on experiments with real-world datasets focusing on user activity recognition.

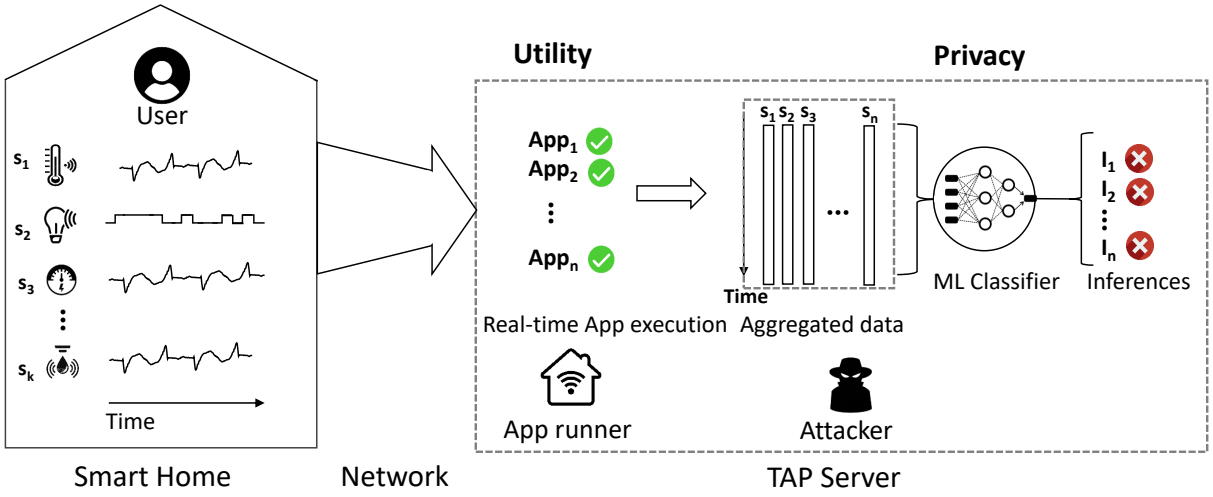


Figure 1: IoT Application Privacy and Utility Concerns in a Smart Home TAP

Our key insight is the use of adversarial machine learning in the face of malicious TAP classifiers. The design of PTAP incorporates a privacy mediator, which serves as a trusted intermediary between smart home devices and the TAP server. The privacy mediator injects targeted perturbations into the data stream, effectively confounding potentially malicious TAP classifiers. Besides data perturbation, the privacy mediator also acts as a filter, distinguishing between legitimate and fake actions sent from the TAP server. Specifically, the privacy mediator leverages local copies of IoT applications to identify fake actions. Fake actions are elicited by emitting copies of fake events to the local copies of IoT applications. The resulting fake actions are then filtered out from the set of actual actions, allowing their removal while passing genuine events back to the IoT devices. By simultaneously employing adversarial machine learning techniques for perturbation generation and the filtering mechanism in the privacy mediator, PTAP offers a robust solution for preserving user privacy and utility in smart home TAPs.

We evaluated PTAP using the real-world Orange4Home dataset [18]. Our experiments revealed that PTAP can confound activity recognition classifiers, reducing their accuracy from 91 % to 6 % through the injection of noise, thereby enhancing privacy. Based on our threat model, the privacy mediator has no knowledge of the attacker classifier. Hence, we tested the effectiveness of PTAP-generated noise against unknown attacker classifiers in a black-box setting. Our experiments demonstrate the transferability of adversarial examples to unknown classifiers, reducing their accuracy from 93% to 4% and from 87% to 8%. Furthermore, our utility test on sample IoT applications demonstrated that PTAP does not alter legitimate TAP operations.

Contributions. The paper offers these contributions:

- **PTAP design:** We present PTAP, a Privacy-preserving Trigger-Action Platform that safeguards user privacy by emphasizing the protection of inferences drawn from aggregated, seemingly nonsensitive data, rather than on individual data points.
- **Adversarial example attacks:** We are the first to employ adversarial example attacks for privacy preservation in the smart home context. Our approach leverages targeted perturbations generated to deceive unknown classifiers developed with malicious TAPs in a black-box setting, effectively preserving user privacy.
- **Utility preservation:** PTAP maintains the utility of the smart home system by injecting perturbations as new event data rather than modifying real data. The privacy mediator filters out fake actions generated from perturbed data, allowing only legitimate actions from real data to be executed in the smart home environment.
- **Seamless integration:** PTAP seamlessly integrates into existing IoT platforms, app development processes, and infrastructure, requiring no modifications. Our solution ensures privacy preservation without disrupting the functionality of IoT applications.

The remainder of the paper is organized as follows: Section II provides an overview of the smart-home architecture, activity recognition, and adversarial example attacks. In Section III, we define and formalize the research problem, and present the threat model and design goals. Section IV introduces PTAP and details the privacy mediator, as well as how we have adopted the FGSM, JSMA, and UAP attacks to our solution. Section V presents experimental results on three attack methods and evaluates their effectiveness against unknown classifiers in a black-box setting. In Section VI, we discuss the strengths, limitations, and future research directions of PTAP. In Section VII, we review related works, and Section VIII concludes. PTAP and all experimental results

are publicly available on GitHub.[1]

II. Background

A. Smart Homes

Smart homes integrate physical processes within homes with digital connectivity to enhance comfort, efficiency, and security. The key components in a smart home comprise (1) IoT devices, (2) Smart home Trigger Action Platforms (TAPs), and (3) IoT applications.

IoT devices. IoT devices are equipped with embedded sensors and actuators that automate physical processes. Sensors gather information about physical states and transmit it as data events to the TAPs. These events are subsequently processed in accordance with IoT applications, triggering the appropriate actions on the relevant devices. Examples of these IoT devices include smart thermostats, light bulbs, presence sensors, and smart door locks.

Smart home trigger action platforms (TAPs). TAPs manage the interactions of devices within the smart home ecosystem, providing app-specific services. They allow users to define reactive applications that respond to sensor data events, generating action commands for respective devices. Prominent examples of TAPs used in smart homes include Apple's HomeKit, OpenHAB, Samsung's SmartThings, and more generic platforms like IFTTT.

IoT applications. IoT applications are automation rules created by users on TAPs. These applications process sensor data and generate action commands for actuators. They enable smart home automations, creating scenarios like "When the motion sensor detects motion at your door, turn on your front door smart light". Examples of IoT applications are SmartApps and automations in SmartThings platform and applets in IFTTT platform.

B. Activity Recognition

Activity recognition aims at identifying user activities within a smart home, using data collected from various sensors embedded in the environment. This task involves applying machine learning models, especially deep learning models [62]. These models are trained on sensor data that are labeled with activities of interest. For instance, a model trained on user data from a smart home environment can precisely identify daily living activities such as cooking, sleeping, showering, watching TV, and leaving home [36].

While user activity recognition has notable benefits, such as aiding in ambient assisted living (AAL) to support elderly people [9], it can also pose significant privacy risks in smart homes. Specifically, an adversary who gains access to this sensor data could potentially deduce sensitive information about the residents of the home by recognizing user daily living activities.

[1]https://github.com/mahmoudaghvami/ptap

C. Adversarial Examples

Despite the significant advances of deep neural networks (DNNs) in tackling complex tasks, they exhibit notable susceptibility to adversarial examples [57]. An adversarial example is a slightly modified input, specifically designed to cause the machine learning model to generate an incorrect output. Given a classifier, f, that consistently produces the correct label for an input vector, x, an adversarial vector x' that is similar to x can be defined as: $f(x) = y,\ f(x') = y',$ and $y \neq y'$.

In this context, the adversarial vector x' is crafted such that it leads the classifier f to produce a different label y' from the correct label y of the original input vector x. In a typical scenario for the generation of adversarial example x', a perturbation vector η is added to the original input x, such that $x' = x + \eta$. The adversarial example x' and the original input x preserve a predefined distance. This ensures that the perturbed input is similar enough to the original one to be considered a valid input, while being different enough to mislead the classifier.

Many techniques have been proposed for generating adversarial examples, among which Fast Gradient Sign Method (FGSM) [23] and Jacobian-based Saliency Map Attack (JSMA) [45] are two of the most popular ones. Additionally, universal adversarial techniques [39] allow for perturbations that are effective against a wide range of inputs, further expanding the potential of adversarial techniques.

While adversarial examples are traditionally viewed as a form of attack against classifiers within adversarial machine learning literature [10, 23, 28, 45, 57], they can also serve as a powerful defense mechanism against an adversary's classifier in the context of smart home TAPs. Research by Papernot et al. [44] shows the transferability of adversarial examples across different classifiers, including classifiers trained with diverse datasets. This suggests that adversarial attacks can be effectively utilized in black-box settings, where the adversary's classifier is unknown.

III. Activity Recognition Protection

This section defines the problem of activity recognition protection in smart home TAPs that preserve utility.

A. Problem Setting

Data sources. Smart homes use a variety of sensors for data capture. Zheng et al. [69] classify smart home data sources into three categories: sensory data, multimedia data, and user interaction data. Privacy concerns often deter smart home users from installing video cameras in private spaces[40], making camera sensors for activity recognition typically unacceptable [59]. Given these insights, our research primarily tackles privacy issues related to sensitive inferences steming from seemingly nonsensitive sensory data and user interaction data, excluding those pertaining to multimedia data. Sensory data in smart homes fall into three groups: numerical (e.g., temperature, humidity), binary (e.g., door status, motion sensor readings), and categorical data (e.g., heater set-points with possible values 16, 19, 21).

Inferences. A smart home trigger-action platform allows for various inferences from sensory and user interaction data, including activity recognition, economic status prediction, and home occupancy identification. Given the vast amount of sensed data in smart homes, manual reviews to extract inferences are impractical and inefficient, especially for attackers seeking to profile users and invade their privacy. Thus, we assume attackers use machine learning algorithms, such as deep neural networks [25], to extract sensitive inferences for user profiling. We specifically focus on user activity classification as a sensitive inference that has the potential to violate user privacy. We consider all classes of user activities to be sensitive in our setting. Additionally, we assume that each time interval corresponds to a single activity class, without any overlapping activities within a given interval.

Privacy. Two levels of privacy pertain to our research problem: Ideal Privacy and Relative Privacy. Ideal Privacy refers to the scenario where all data samples that are deemed sensitive according to user privacy preferences are misclassified by the attacker's classifier. This level of privacy ensures that none of the sensitive information is successfully inferred by the attacker. However, achieving ideal privacy in real-world scenarios can be challenging.

Relative Privacy is a level of privacy preservation in which the attacker's classifier can correctly detect a limited number of sensitive samples. The disclosure budget, determined by the user, sets the acceptable threshold for the percentage of correct classifications by the attacker's classifier. Any correct classification above this threshold is considered a privacy violation. This approach offers a practical and realistic means of privacy preservation.

Utility. The primary utility for users in a smart home TAP is the correct execution of their IoT applications. Utility is of paramount importance as any compromise can have irreversible effects and potentially endanger the safety of physical environment. In this paper, our proposed solution preserves the correct execution of IoT applications.

B. Threat Model

Following previous research [14, 15, 16], our threat model considers an untrustworthy TAP. In this setting, the TAP is either compromised or malicious, and employs deep-learning-based classifiers to infer sensitive information from user data. We assume a malicious TAP that has developed an activity recognition classifier, which assigns activity class labels to each time interval. However, our solution is independent of the specific classifier and can be applied to other types of classifiers, e.g., economic status prediction or home occupancy detection, based on smart home sensory data and user interactions. Because our adversary is a malicious TAP, we have no knowledge about the deep-learning model. We therefore view the adversary classifier as a black box.

Our threat model does not consider attacks such as denial of service, automation rule integrity, and action integrity attacks. Additionally, we assume that a single data is nonsensitive for the user and focus on sensitive inferences derived from aggregated user data. We also assume that IoT devices, smart home hubs, and communication channels between the TAP and the smart home are trustworthy. In fact, the TAP represents the most attractive target for adversaries to compromise, as it aggregates data from multiple users.

C. Design Goals

To achieve privacy and utility in the smart home TAP environment, while minimizing changes to the TAP architecture, we formulate our design goals as follows:

Privacy goal. We aim to preserve relative privacy by limiting the number of correct classifications made by the attacker's classifier on data samples. The disclosure budget, defined by the user, sets the acceptable threshold for the percentage of correct classifications. Any correct classification above this threshold is considered a privacy violation.

Utility goals. To ensure the practicality and effectiveness of our solution, we have defined the utility goals:

1. Preservation of existing TAP architecture: We aim to keep the existing TAP software stack, smart home hubs, and IoT devices intact. Making changes or adding new software plugins to these entities can present significant barriers and complexities in real-world implementations. Our approach focuses on empowering users to preserve their privacy locally, without requiring changes in commercial TAPs or IoT devices.

2. Preservation of IoT application execution: We ensure that data perturbation does not compromise the correctness of IoT application execution. The objective is to uphold the functionality of IoT applications even with injected noise. This means that all expected IoT applications should run without any disruptions, and no unintended or extraneous IoT application actions should be executed due to injected noise.

3. No impact on IoT application development: Our solution aims to integrate seamlessly with existing IoT application development processes, allowing users to continue using their current IoT applications without modifications or adaptations.

D. Formalization of Problem Definition

In our setting, we use a set S of m sensors, with each sensor written as $s_i \in S$. A data event, denoted as $e_i = (s, v, t)$, represents a sensor reading. Here, $s \in S$ refers to the sensor from which the reading originates, $v \in \text{Domain}(s)$ is the value of the sensor reading in its domain, and $t \in T$ is the time of the event.

Definition 1: **(Aggregated data samples)**
$X_i \in X, X_i = [t_i, x_1, x_2, x_3, ..., x_m]$
$X_i.se = [x_1, x_2, x_3, ..., x_m]$, $X_i.t = t_i$
Here, X_i represents an element of the set X and denotes an aggregated data sample corresponding to the time window ΔT starting at t_i. X_i consists of a start time t_i and m elements, with each element x_j (where $1 \leq j \leq m$) representing the value of the last read event from the respective sensor. If there is an updated value from the corresponding sensor within the ΔT time window, the updated value is assigned to x_j. If there is no updated event from the corresponding sensor within the

ΔT time window, then x_j is set to the previous value that has been read from the corresponding sensor.

Definition 2: **(Data set)**
$D = \{(X_1, y_1), (X_2, y_2), ..., (X_n, y_n)\}$, where $X_i \in X$ and $y_i \in L$
D represents a set consisting of n pairs of X_i and y_i. Each pair in D consists of X_i, which represents the smart home aggregated data sample, and y_i, which represents the corresponding user activity class label. Importantly, there is only one activity per data sample. The data set D is dynamic and continually growing, as new smart home samples and user activity class labels are added. Each y_i is an element of L, the set encompassing all possible user activity classes.

Definition 3: **(TAP deep learning classifier model)**
$\mathrm{f}(\cdot) : X \to L$
We define the TAP deep learning classifier model as a function $\mathrm{f}(\cdot)$ that maps an input space X (n-dimensional vectors representing smart home data samples within a specific time window) to an output space L (set of labels corresponding to the predicted activity class for the given time window). Essentially, this formalizes the TAP model's role in utilizing deep learning techniques to predict activity classes based on smart home data samples.

Definition 4: **(Ideal privacy)**
$\forall\ (X_i, y_i) \in D$, if $y_i \in$ Sensitive_Activities, then
$\exists\ X_i'$, $\mathrm{f}(X_i') = y_i'$ and $y_i \neq y_i'$ and $||X_i' - X_i||_p \leq \epsilon$

This definition asserts that for every pair (X_i, y_i) in the dataset D, where y_i is a sensitive activity, there exists a perturbed version X_i' that the TAP deep learning model misclassifies as y_i' (with $y_i \neq y_i'$). Additionally, X_i' must be close to X_i within a specified ϵ distance in the p-norm sense.

Definition 5: **(Relative privacy)**
$\forall\ (X_i, y_i) \in D$, if $y_i \in$ Sensitive_Activities, then
$\exists\ X_i', Pr(\mathrm{f}(X_i') \neq y_i) \geq 1 - b$ and
$||X_i' - X_i||_p \leq \epsilon$
where b is the disclosure budget.

This definition states that for every pair (X_i, y_i) in the dataset D, there exists a perturbed version X_i' such that the TAP deep learning model misclassifies as y_i' (with $y_i \neq y_i'$) with a probability of $1 - b$, where b is the disclosure budget defined by the user. Moreover, X_i' should remain close to X_i within a specified ϵ distance.

Definition 6: **(Utility function)**
$U : E^n \times T \times T \to P(A)$
The utility function U maps each sequence of events, e. g., "$se \in E^n$ occurring within a time-interval $[t_1, t_2]$", to a subset of actions from A. These actions denote the result of IoT application executions for the given sequence of events se at this time interval.

Definition 7: **(Utility preservation)**

$$\forall\ (X_i, y_i) \in D, U(X_i.se, X_i.t, X_i.t + \Delta T) = U(X_i'.se, X_i'.t, X_i'.t + \Delta T),$$

where X_i' is a perturbed sample data of X_i
For each data sample $X_i \in D$, the utility is preserved if the actions generated by the real events are also generated by the perturbed events (the mix of real and injected fake events), and no spurious actions are generated by fake events. Without any precaution, we may have extra actions that are triggered by the fake events. Therefore, we filter the fake actions to keep only actions from real events. In this way, soundness and completeness of IoT applications (utility) is guaranteed.

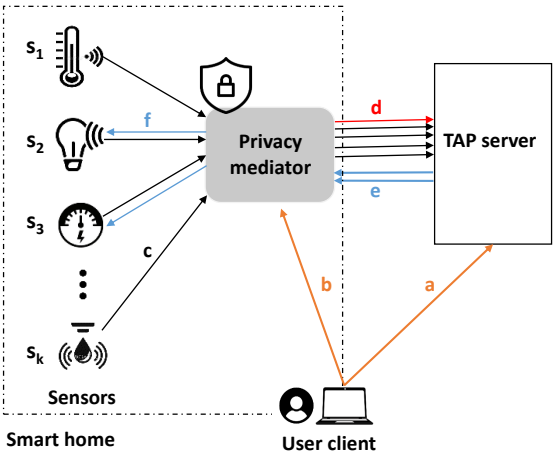


Figure 2: Privacy-preserving Trigger Action Platform Architecture

IV. PTAP Platform

In this section, we discuss the PTAP architecture and detail our technical solution to employ adversarial example attacks to protect users' sensitive information.

A. Architecture Overview

Our Privacy-preserving Trigger-Action Platform (PTAP) is designed to address privacy concerns while maintaining the utility of smart home TAPs, without requiring any modifications to existing TAPs, IoT hubs, or IoT application development practices. The privacy mediator is the main module in our architecture that generates the perturbation (see Section IV-C) and filters fake actions (see Section IV-D) based on the received data stream. Figure 2 displays the PTAP architecture, illustrating the interactions among the IoT devices, privacy mediator, TAP server, and user client.

The overall process comprises six steps (see Figure 2):

a) IoT application setup: By using the TAP interface, the user set up a new IoT application. This includes selecting a trigger, and an action, defining the fields for the trigger and action, and constructing the logic of the application.

b) Sharing IoT applications and privacy preferences with the mediator: During the application setup phase, the user shares the combination of triggers, actions, and application logic that form an IoT application with the privacy mediator through the user client. Additionally, the user defines privacy preferences (see Section IV-B) by selecting sensitive activity classes and communicating them to the privacy mediator device.

c) Transmission of new events from the IoT device to the privacy mediator: IoT devices periodically generate data events

(stemming from environmental changes within the house, such as temperature fluctuations, or user activities and interactions with devices, like activating a smart light) and forward them to the privacy mediator.

d) Injection of perturbation and forwarding to the TAP server: Based on user privacy preferences, the privacy mediator generates targeted perturbations for event data (as explained in Section IV-C) and sends the perturbed events to the TAP server.

e) Execution of IoT applications and sending of action commands: The TAP server processes the received perturbed events and, upon executing the IoT applications, dispatches action commands back to the privacy mediator.

f) Filtering of fake action commands and transmission of real actions to devices: The privacy mediator filters out fake action commands generated from the perturbed data and sends the real action commands to the IoT devices for execution.

By following this process, PTAP enables users to control their sensitive information and safeguard their privacy while still benefiting from the functionality of smart home TAPs.

B. User Privacy Preferences

As IoT environments continue to grow, privacy by design principles have not been sufficiently integrated into the majority of commercial and open-source smart home platforms. Consequently, users lack the ability to specify their privacy preferences [29]. We aim to bridge this gap by defining privacy preferences from the viewpoint of potentially sensitive user activities, simplifying privacy management for non-expert smart home users. In PTAP, users select their privacy preferences from their potential user activities. In line with existing works on Human Activity Recognition (HAR) systems [18, 36, 37], we have chosen the list of these activities, as displayed in Table I. This empowers users to control their sensitive activies, enhancing their ability to safeguard their personal privacy.

C. Data Perturbation via Adversarial ML

PTAP protects users' privacy by preventing sensitive inferences through the use of adversarial example attacks. Adversarial example attacks involve an attacker compromising the accuracy of a classifier by injecting a small perturbation into the input data, leading to incorrect classification results. In PTAP, we employ a similar strategy by injecting minor perturbations into smart home data, thereby deceiving malicious TAP classifiers attempting to identify users' daily living activities, which would otherwise violate their privacy. As noted in Section II-C, the majority of research on adversarial example attacks has focused on image processing, and there are unique challenges associated with applying these attacks to smart home contexts for privacy preservation:

Challenge 1 Modification of smart home data is not allowed, as the utility derived from executing IoT applications is critical and essential for smart home users. In fact, the compromise of utility in a smart home could result in safety violations in the physical environment, such as leaving a smart lock on the main door open or a smart oven turned on. Consequently, each individual data event sent to the TAP must remain unaltered.

Challenge 2 Unlike image processing contexts where all features share the same domain (e. g., 0 to 255 for all image pixels), smart homes involve different sensor types with entirely distinct domains, including binary sensors, categorical sensors, and numerical sensors. Therefore, any perturbations must occur within each feature's specific domain.

Challenge 3 Unlike image processing scenarios where the image is provided from the beginning, smart home data is time series data that grows over time, influenced by both user activities and environmental changes. Since the data content is not static from the start, the dynamic nature of user data necessitates that perturbations are both adaptive and responsive to changes in the data as it evolves.

To address these challenges, PTAP incorporates three adversarial example attack methods: FGSM, JSMA, and Universal Adversarial Perturbation (UAP). Each method employs distinct strategies: FGSM focuses on minimizing the amount of perturbation, JSMA aims to minimize the number of perturbed sensors, while UAP strives to generate perturbations that are universally applicable across multiple inputs. We evaluate the effectiveness of each method in preserving user privacy in our experiments in Section V. To ensure the preservation of utility, PTAP injects only new perturbation points into the data streams to the TAP, while keeping the real data points unaltered. Moreover, to ensure that perturbations fall within each sensor's domain, we adapt the generated perturbation for each sensor type accordingly. For numerical sensors, we apply the sensor's range; for categorical sensors, we choose the nearest possible value to the calculated perturbation; and for binary sensors, we toggle the previous value if a perturbation is needed.

1) Adversarial Setting:

Untargeted adversarial examples. In our privacy preservation framework, untargeted adversarial examples are used to subtly perturb the TAP classifier. These perturbations aim to disrupt the accuracy of the TAP's activity recognition classifier without forcing a specific incorrect classification label, thereby maintaining user privacy.

Black box setting. In line with our threat model, the TAP classifier is considered a black box for the privacy mediator, meaning the privacy mediator has no knowledge of the machine learning algorithm's architecture and hyper-parameters. Utilizing the transferability feature of adversarial example attacks, we compute perturbations based on a surrogate model within the privacy mediator. The results of our transferability tests, which detail the effectiveness of these perturbations, are discussed in Section V-C.

L^∞ and L^0 metrics. To measure the similarity between the perturbed vector x' and the original vector x, we use two distance metrics that have been extensively used in adversarial example attacks. The L^p distance is $\|x - x'\|_p$, where the p-norm $\|.\|_p$ is defined as:

$$\|x\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$$

The L^∞ distance measures the maximum change to any of the coordinates:

$$\|x - x'\|_\infty = \max(|x_1 - x'_1|, \ldots, |x_n - x'_n|)$$

For vectors, we can imagine there is a maximum budget and each feature is allowed to change up to this limit, with no restriction on the number of features that can be modified [10]. In our work, we employ the L^∞ norm for the FGSM algorithm and UAP, focusing on the maximum perturbation allowed for any single sensor.

The L^0 distance measures the number of coordinates i for which $x_i \neq x'_i$. In other words, the L^0 distance corresponds to the number of features that have been altered in a vector [10]. In our work, we use the L^0 norm for the JSMA algorithm, aiming to minimize the number of perturbed sensors.

2) Fast Gradient Sign Method (FGSM): The Fast Gradient Sign Method (FGSM) [23] is a popular and computationally efficient adversarial attack technique that aims to maximize the adversarial perturbation by manipulating the input data. By utilizing the L^∞ norm, FGSM measures the maximum change to any of the features. Algorithm 1 presents the adopted FGSM. Initially, the target classifier, denoted by f, is the malicious TAP classifier. Subsequently, a target input data vector, X_i, is the smart home data within a specific time window.

In the first step, the algorithm masks X_i using the set of updated sensors, $\texttt{US}$, within a Δt time window. This step masks the features related to these updated sensors, preventing them from undergoing any alteration. As a result, the algorithm produces a masked vector, X_i^m.

Next, the algorithm computes the gradient of the loss function concerning the input data, $\text{sign}(\nabla_x \mathbb{J}_\mathrm{f}(X_i^m, \mathrm{f}))$. In order to generate a perturbation, FGSM multiplies the sign of this gradient by a small constant, ϵ. This perturbation is then added to the masked input data vector to produce a perturbed input data vector, X'_i.

Lastly, the algorithm conducts a domain adaptation on the perturbation. This critical step ensures the perturbation remains within the allowable domain of each feature, thereby maintaining the validity of the data for TAP.

3) Jacobian-Based Saliency Map Attacks (JSMA): The Jacobian-based Saliency Map Attack (JSMA) [46] is a targeted adversarial attack that modifies the most critical features in the input data, hence minimizing the number of altered features. This approach employs the L^0 norm to quantify the number of non-zero elements in the perturbation vector, emphasizing the importance of minimizing feature perturbation. In the smart home context, as presented in Algorithm 2, we consider f as the malicious TAP classifier and X_i as the target input data vector, denoting the vector of smart home data within a specific time window.

The algorithm begins by masking X_i with the set of sensors updated within the Δt time window, referred to as $\texttt{US}$. This step leads to the creation of the masked vector, X_i^m, which keeps the associated features unaltered.

Algorithm 1 Adopted FGSM

Input: Data set D containing data sample X_i, activity recognition classifier f, maximum perturbation ϵ, number of iterations n, batch size $\texttt{batch\_size}$, sensor domains $\texttt{SD}$, set of updated sensors $\texttt{US}$ within Δt time window
Output: Perturbed data sample X'_i

for $t = 0$ to n **do**
  Select a batch of $\texttt{batch\_size}$ data samples from D, denoted by X^{batch}
  for each sample X_i in X^{batch} **do**
    $X_i^m = \text{Mask}(X_i, \texttt{US})$
    $\eta = \epsilon \cdot \text{sign}(\nabla_x \mathbb{J}_f(X_i^m, f))$
    $X'_i = X_i^m + \eta$
    $X'_i = \text{Domain\_adoption}(X'_i, SD)$
  end for
end for
return X'_i

Our adversarial strategy primarily follows an untargeted approach; however, JSMA necessitates the designation of a target class for the execution of the attack. To meet this requirement, we select a target class randomly for each iteration. During each of these iterations, if the classifier's output does not match the target classification, y_{target}, and the number of perturbations is less than the predefined maximum, k, the algorithm computes the gradient of the classifier's output with respect to the input data. This gradient informs the saliency map which ranks each feature's importance.

Using the saliency map, the top k features with the highest saliency scores are selected for perturbation. The algorithm perturbs these features and re-evaluates the perturbed input data vector, X'_i, against the classifier. If the classifier's output aligns with the target classification or the maximum number of perturbations has been attained, the algorithm applies a domain adoption function on the perturbation. This step assures that the perturbation is within the acceptable range of sensor values.

4) Universal Adversarial Perturbation (UAP): Universal Adversarial Perturbation (UAP) [39] is a type of adversarial attack that aims to generate perturbations that are universally applicable across multiple inputs. Unlike FGSM and JSMA, which produce input-specific perturbations, UAP seeks to create a single perturbation vector that, when applied to any input, causes the classifier to produce incorrect prediction. This unique characteristic enables the generation of a one-time perturbation that can be universally applied. This is particularly valuable for smart home data that initially is neither complete nor fixed, but it continuously evolves over time due to user activities and environmental changes.

Within the PTAP framework, as depicted in Algorithm 3, the UAP is initially computed using a subset of the data. This universal perturbation is then applied to subsequent data

Algorithm 2 Adopted JSMA

Input: Data set D containing data sample X_i, activity recognition classifier f, set of class labels L, target classification y_{target}, maximum perturbations k, number of iterations n, batch size `batch_size`, sensor domains `SD`, set of updated sensors `US` within Δt time window

Output: Perturbed data sample X'_i

for $t = 0$ to n **do**
  Select a batch of `batch_size` data samples from D, denoted by X^{batch}
  for each sample X_i in X^{batch} **do**
    y_{target} = randomly select an element from L
    $X_i^m = \text{Mask}(X'_i, \texttt{US})$
    $X'_i = X_i^m$
    while $(\text{f}(X'_i) \neq y_{\text{target}}$ and number of perturbations $< k)$ **do**
      gradient $= \frac{\partial \text{f}(X'_i)}{\partial X'_i}$
      $S = \text{SaliencyMap}(\text{gradient})$
      $F = \text{TopFeatures}(S)$
      $X'_i = \text{Perturb}(X'_i, F)$
    end while
    X'_i = $\text{Domain\_adoption}(X_i^3 2m, SD)$
  end for
end for
return X'_i

Algorithm 3 Adopted Universal Adversarial Perturbation (UAP)

Input: Data set D containing data sample X_i, classifier f, L^∞ norm of the perturbation ξ, desired accuracy on perturbed samples δ, set of updated sensors `US` within Δt time window

Output: Universal perturbation vector v

Initialize $v \leftarrow 0$
while Err$(X_v) \leq 1$ - δ **do**
  for each data point x_i in D **do**
    $x_i^m = \text{Mask}(x_i, \texttt{US})$
    if $\text{f}(x_i^m + v) = \text{f}(x_i^m)$ **then**
      Compute the minimal perturbation that sends $x_i^m + v$ to the decision boundary:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ s.t. } \text{f}(x_i^m + v + r) \neq \text{f}(x_i^m)$$

$$v \leftarrow P_{p,\xi}(v + \Delta v_i)$$

$$v \leftarrow \text{Domain\_adoption}(v, \texttt{SD})$$

    end if
  end for
end while
return v

samples. Given that UAP is universally applicable, this perturbation is particularly beneficial in scenarios where we seek real-time perturbation, without imposing delays in transferring the events of a time-window. In this scenario, we apply the pregenerated universal noise in real-time while simultaneously sending new real data events to TAP.

D. Utility Preservation

A key requirement in applying adversarial example attacks in smart home data is the importance of utility. As mentioned before, smart home utility directly affects the physical environment; therefore, compromising the utility in some cases can cause safety violations. PTAP ensures that no real data is perturbed by only injecting new events as perturbations while filtering out all fake actions received from the TAP.

When the privacy mediator receives events from an IoT device, it forwards them unchanged to the TAP server. For FGSM and JSMA attacks, PTAP computes the perturbation at the end of each time window and injects it as new data to the TAP. For UAP, PTAP injects a pre-generated universal adversarial example in real time. As a result, the TAP receives perturbed data and executes IoT applications based on it. Since the original data remains unperturbed, the real actions are preserved.

However, the TAP will also execute IoT applications based on generated fake data, and the mediator will receive the result of these executions (fake actions). To address this challenge, we employ a filtering mechanism in the privacy mediator that detects and filters the received fake actions while allowing the real actions to pass. When the mediator creates fake event data through adversarial techniques and sends it to the TAP, the filtering mechanism determines if the generated fake data is associated with the input trigger of any installed IoT applications.

The privacy mediator uses the local copies of IoT applications to generate all potential actions locally. It therefore knows what actions are generated by fake events. When receiving the actions from TAP, the privacy mediator compares the received action with the list of expected fake actions. Through this mechanism, the privacy mediator can filter out all of the fake actions and only pass the real actions. In cases where the generated fake data causes an IoT application to trigger an action, the action obtained by executing the application locally with the generated fake data matches the action from the TAP, allowing it to be filtered.

There are two key points that help the filtering mechanism to achieve good performance and accuracy. First, the TAP should preserve the order of execution of IoT applications and forward responses based on the order of event data that has been received [22]. This order can help the mediator to predict the expected order of fake actions from the TAP. However, if the order of received actions from TAP changes, e.g., due to network interruption or discrepancy in computation time of different IoT applications, the mediator will still filter out the correct fake action. This is because the mediator runs the IoT application with the fake data locally and has the exact fake action as the result of local execution.

Second, for the sake of the filtering mechanism, the mediator only needs to run the IoT applications related to the

generated fake data, and there is no need to execute a local copy of the applications with the real data. With this assumption and the fact that we minimize the number of required perturbed features with the JSMA algorithm, we only have a small number of perturbed features during each time window (for example, less than 10 percent of all possible IoT devices) that need possible local execution for detecting the fake action. This significantly reduces the local execution overhead outside of the TAP server.

E. Running example

Here, we describe a running example application, which progresses through steps of our architecture process (see Section IV-A). We present abridged data vectors here; the complete data vectors of our example are available in our public repository.

Step a: IoT application setup. Our example IoT application involves the logic "When the voltage rises above the specified threshold, turn off the appliance(s) connected to the selected plug". The user sets up this application by selecting the trigger based on kitchen cooktop voltage data, choosing the action to turn off the cooktop connected to the selected plug, and defining a threshold of 240 volts for the IoT application.

Step b: Sharing IoT applications and privacy preferences with the mediator. The user shares the IoT application with the privacy mediator through the user client. Additionally, the user defines privacy preferences by selecting specific activity classes from the provided list of potential activities as sensitive. In our example, the user has identified the activity classes 'Cooking,' 'Entering home,' and 'Leaving home' as sensitive.

Step c: Transmission of IoT events to the privacy mediator. IoT devices periodically generate data events representing environmental changes and user interactions in the smart home. These events are then forwarded to the privacy mediator for processing.

Step d: Injecting perturbations. The original input data to step d comprises sensor data points, including cooktop voltage readings. In this example, the aggregated data reading of time window t is (0, 0, 21, ..., 238.25, ...). In the current time window, the privacy mediator receives updated data for some sensors and aggregates them with the last received values of other sensors. This process results in a 196-element vector that presents the updated status of the smart home, reflecting the user's current activity of cooking. The last updated value for the cooktop voltage is 238.25, which was received in the previous time window.

During this step, the running JSMA attack method is employed to perturb the data. With the gamma attack parameter set to 0.01, which means 20 sensor data points will be perturbed among 196 sensor data points. After perturbation, the 20 sensor data points have been perturbed and sent to the TAP, resulting in the aggregated data vector for time window t as (0, 0, 21, ..., 240.57, ...), where the perturbed value for cooktop voltage has increased to 240.57 and has been sent to the TAP.

The TAP receives data points and attempts to predict the user activity based on the received data. Since adversarial perturbation has been injected, the TAP classifier is misled in predicting user cooking activity, thereby preserving user privacy.

Step e: Executing the IoT application.

In this step, based on the normal functionality of TAP, the IoT applications are executed with the input of received data points. Considering our running example, the application involves monitoring cooktop voltage and taking action when it surpasses a specified threshold (240.0). The last received data for the cooktop voltage is 240.57. In our example case, when the cooktop voltage exceeds the threshold of 240 volts, the application triggers the action of turning off the cooktop connected to the selected plug, which is considered an unwanted fake action.

Step f: Filtering fake actions.

Following the execution of the IoT application with the perturbed data, the privacy mediator filters out any fake action commands generated from the perturbed data. In our example scenario, this means identifying and disregarding the fake action commands of turning off the cooktop due to the perturbed cooktop voltage. The privacy mediator has access to the IoT applications and runs the applications that have a trigger of perturbed data. In our example, the application will be executed with perturbed data input of 240.57 volts and will yield a fake action result of "turn off the cooktop connected to the selected plug". Then the privacy mediator will wait to receive the predicted fake action by the TAP and then drop it. By effectively filtering out fake actions, the privacy mediator ensures that only genuine actions, based on real data, are executed within the smart home environment.

V. Evaluation

We evaluate PTAP by addressing the following research questions:

RQ1: How successful is PTAP in preserving user privacy by misleading TAP classifiers and hiding sensitive inferences? How do FGSM, JSMA, and UAP attack methods compare in this context?

RQ2: How effective is PTAP in a black-box setting against an unknown classifier with different architecture and hyperparameters?

RQ3: How effective is PTAP in preserving user utility in terms of IoT application execution?

RQ4: What is the computational overhead and network cost associated with PTAP?

To answer these research questions, we established a comprehensive evaluation process. In the role of a malicious TAP, we developed three different activity recognition classifiers that represent the TAP's intent to infer user activities. On the other hand, we designed a privacy mediator leveraging FGSM, JSMA, and UAP methods to generate adversarial perturbations and perturb smart home data. Our evaluation of the efficacy of deceiving the TAP classifier by injecting targeted noise

provides insights into the comparative effectiveness of these methods.

For RQ2, we evaluated the transferability of adversarial examples generated by the privacy mediator on two unknown black-box classifiers of the malicious TAP. In this setting, we assumed the privacy mediator has no knowledge about the attacker's classifier, and it is up to the attacker to choose an effective unknown attack model against the privacy mediator. This was done to emphasize the generality of PTAP's approach, regardless of the adversary's choice in user-activity classifiers. Specifically, we assessed how successfully PTAP could operate against classifiers with different architectures and hyperparameters, ensuring that PTAP-generated perturbations are robust against a variety of potential attack models.

For RQ3, we analyzed the ability of PTAP to preserve user utility in terms of IoT application execution. We evaluated this by comparing the functionality and performance of selected IoT applications with and without the presence of PTAP.

Lastly, for RQ4, we evaluated the computational and network overheads of the PTAP. Two primary metrics were considered: the time taken for each attack execution, and the number of events injected for each run.

A. Experimental Setup

Dataset: For our experiment, we used the Orange4Home dataset [18], which includes approximately 180 hours of recorded daily living activities of a single smart home user over four consecutive weeks of workdays, from 8 AM to 6 PM. The dataset was collected in a smart apartment equipped with a total of 236 sensors distributed across various rooms, including the entrance, kitchen, living room, toilet, staircase, walkway, bathroom, office, and bedroom. These recordings were made with the OpenHAB platform [41], a prominent open-source TAP in the IoT community. The sensors were designed to measure different parameters, such as temperature, humidity, CO2 levels, noise, lighting, energy consumption, and water consumption, along with switch devices and specific sensors like fridge and oven sensors in the kitchen and bed pressure sensors in the bedroom.

The full dataset contains 745,782 events. Each event is represented by three attributes: timestamp, item name, and value, indicating an update from an IoT device within the house. We excluded data from 40 global sensors; a majority of these sensors were related to external weather conditions and were not directly tied to the internal smart home environment sensors. We further preprocessed the dataset and created a merged data table with one row of data per minute, assuming that the TAP classifier needs to aggregate data for the activity recognition task. As part of the preprocesisng, we filled any missing data readings during each sample with the sensor's last updated value.

Table I details the Orange4Home dataset, presenting the number of instances in each of the 24 activity classes for the first iteration, where the initial three weeks form the training set and the fourth week makes up the testing set. Apart from these 24 activity classes, we introduced an "inter-activity" class to denote the indefinite periods between two activity classes.

Activity recognition classifier: We designed a 1D-CNN activity recognition classifier to identify the activity class label for each time window in the smart home sensor data. To reflect common practices in activity recognition, we segmented the data flow into regular 60-second time windows as suggested by Kasteren et al. [60], a practice also endorsed by several other studies [24, 34]. Our classifier then assigned an activity label to each time window. To evaluate our model, we applied 4-fold cross-validation, as we have four weeks of recorded data. In each iteration, three weeks were used for training, and one week was reserved for testing. This approach aligns with the previous study that introduced an activity recognition classifier for the Orange4Home dataset [36]. Our classifier achieves an average accuracy of $91.72\,\%$ for 4-fold cross-validation.

IoT applications: To assess PTAP's impact on user utility, we selected five IoT applications, each representing a distinct category of smart home sensors in our dataset. These applications included smart lighting, temperature monitoring, door sensing, power consumption tracking, and an environmental sensing application.

Hardware: The entire evaluation process was conducted on a computer equipped with 31 GiB of RAM and an Intel(R) Xeon(R) CPU E5-2630 v2 with 24 cores operating at 2.60 GHz, ensuring seamless execution of the evaluation tasks.

Table I: Activity class distribution in the first iteration of the Orange4Home dataset, split into three weeks of training and one week of testing.

No.	Activity Classes	Training	Testing	Total
1	Bathroom—Cleaning	7	2	9
2	Bathroom—Showering	258	71	329
3	Bathroom—Using_sink	92	26	118
4	Bathroom—Using_toilet	27	8	35
5	Bedroom—Cleaning	6	0	6
6	Bedroom—Dressing	22	9	31
7	Bedroom—Napping	292	108	400
8	Bedroom—Reading	350	74	424
9	Entrance—Entering	28	5	33
10	Entrance—Leaving	12	5	17
11	Kitchen—Cleaning	5	3	8
12	Kitchen—Cooking	170	39	209
13	Kitchen—Preparing	24	6	30
14	Kitchen—Washing_dishes	85	21	106
15	Living_room—Cleaning	16	4	20
16	Living_room—Computing	351	103	454
17	Living_room—Eating	173	46	219
18	Living_room—Watching_TV	401	92	493
19	Office—Cleaning	7	3	10
20	Office—Computing	5271	1779	7050
21	Office—Watching_TV	217	44	261
22	Staircase—Going_down	14	2	16
23	Staircase—Going_up	14	5	19
24	Toilet—Using_toilet	12	4	16
25	interActivity	64	92	156
Total		**7966**	**2536**	**10502**

B. Privacy Tests

To answer RQ1, we subjected the classifier to adopted FGSM, JSMA, and UAP attacks. The FGSM attack was

conducted with a batch size of 32 and various epsilon values ranging from 0.01 to 0.5. Figure 3a illustrates the classifier's accuracy under the FGSM attacks with different epsilon values. During this attack, we preserved binary and categorical sensors without any change, aiming to evaluate the effectiveness of reducing the accuracy of a malicious TAP classifier by adding a small amount of perturbation (less than epsilon) to the numerical sensors. The results demonstrate that by adding a perturbation with a maximum value of 0.15 to the smart home data, the TAP's classifier accuracy falls to 16.73 percent.

The JSMA attack also was conducted with a batch size of 32 and various gamma values ranging from 0.01 to 0.51. The gamma parameter represents the fraction of features allowed to be perturbed. The objective of this attack was to illustrate the effectiveness of minimizing the number of features chosen for perturbation in reducing the malicious TAP classifier's accuracy. Figure 3b illustrates the classifier's accuracy under the JSMA attack with different gamma values. The results reveal that a gamma value of 0.1, which permits perturbation of 20 sensors in our case study, leads to a decrease in the malicious TAP classifier's accuracy to 29.55 percent. Since the original JSMA is a targeted attack, we executed the algorithm twenty-five times with twenty-five randomly generated target vectors. Figure 3b displays the average accuracy of these runs and the variance of accuracies for each attack parameter. This diagram highlights an overall reduction in accuracy from 91 % to below 10 %.

The UAP attack was also executed with various epsilon values, ranging from 0.001 to 0.005. Figure 3c illustrates the classifier's accuracy when subjected to the UAP attack, showing a significant decrease in accuracy from 91 percent to 6.7 percent. Given that the UAP perturbation is pre-generated, we distributed it uniformly, taking into account the distribution of real data events. This distribution ensures that the TAP has no opportunity to detect the UAP's distribution pattern. Despite the image classification context in which UAP typically has weaker results in comparison with other nonuniversal attack techniques, the simpler nature of time series data may lend itself to more effective exploitation by UAP methods, as evidenced by some of the datasets presented in [49].

In summary, each adversarial attack method—FGSM, JSMA, and UAP—has its unique characteristics and implications for privacy preservation and classifier disruption. JSMA proves advantageous in scenarios that demand minimal sensor perturbation, preserving privacy by selectively altering data features. FGSM stands out for its simplicity and, requiring minimal computational time, effectively diminishes classifier accuracy. UAP demonstrates remarkable efficacy, drastically reducing classifier accuracy. However, its reliance on pre-generated perturbations necessitates prior access to data samples, which could constrain its immediate applicability. While all three attack methods are effective, the choice of attack method depends on a trade-off between the number of perturbed sensors and the amount of perturbation. In general, we recommend UAP, but if there are severe resource constraints, FGSM is an alternative.

C. Transferability of PTAP

Adversarial examples exhibit transferability across various classifiers and training sets. Due to the limited availability of data for testing different training sets, we focused on evaluating transferability among other classifiers in order to answer RQ2. To achieve this, we trained three target classifiers and used the adversarial examples, which were generated by our original model, as perturbations.

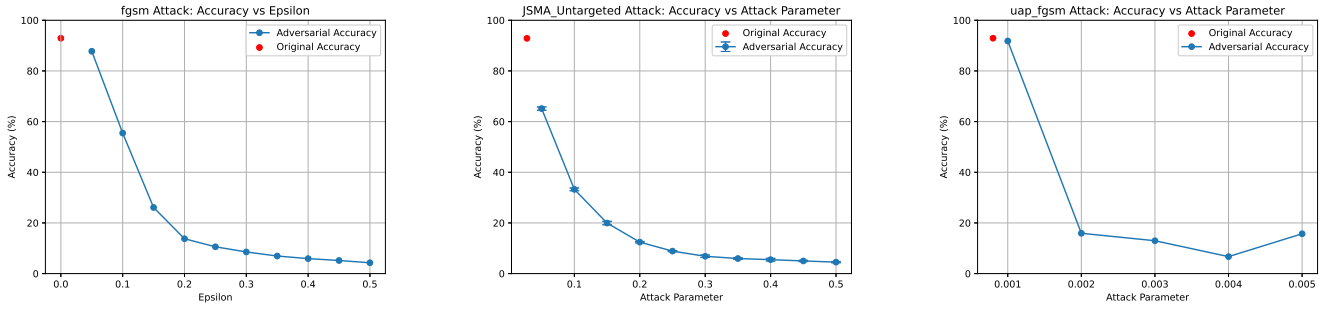
We selected three classifiers for our evaluation: a Multilayer Perceptron (MLP) and a Long Short-Term Memory (LSTM) classifier based on the architectures proposed by Mihoub et al. [36] for activity recognition in the Orange4Home dataset, and a Deep Convolutional Neural Network (DeepCNN) developed by ourselve. The MLP model incorporates two hidden layers besides the input/output layers. The LSTM model consists of three layers in addition to the input/output layers, which include one LSTM layer, a dropout layer to mitigate overfitting, and a dense layer. Our DeepCNN model, employs a series of convolutional and max-pooling layers followed by dense layers, designed to capture temporal features in the data effectively.

While we adhered to the architecture from Mihoub et al. [36] for the MLP and LSTM classifiers, the DeepCNN was independently constructed by our team. Adhering to the principle of independence in PTAP's design, we had no prior knowledge of the parameters of these target classifiers within the privacy mediator while generating the perturbations. We subsequently utilized the adversarial examples produced by our original model to assess the transferability of PTAP across unknown classifiers with varied architectures. Table II presents the accuracy of the target classifiers against benign data and the PTAP-generated perturbations with different attack methods. The results demonstrate that PTAP perturbation is transferable among various DNN architectures, indicating its effectiveness in preserving privacy across different models in black-box settings.

D. Utility Tests

In order to address RQ3, we selected five IoT applications to assess the preservation of user utility in presence of PTAP. Each of these applications was chosen to represent a different device category from our dataset. The chosen applications are as follows:

1) Air conditioning: If the temperature in the living room rises above 20 degrees Celsius, turn on the living room air conditioner [58].
2) Power/water consumption: If the power consumption of the kitchen exceeds 80 W, turn off the washing machine [54].
3) Presence: If the presence sensor in the bedroom detects presence, turn on the bedroom light [50].
4) Door/smart lock: If the entrance door is opened, turn on the entrance light [43].
5) Environment: If the humidity in the living room exceeds 40, turn on the living room air conditioner [19].



(a) (b) (c)

Figure 3: Classifiers' accuracy under various attacks. (a) FGSM attacks with varying epsilon values. (b) JSMA attack with different gamma values. (c) Universal perturbation attack with different epsilons values.

Table II: Assessing the transferability of attacks across MLP, LSTM, and DeepCNN target classifiers

Attack Method	**Attack Parameters**	**Overall Accuracy**		
		MLP	**LSTM**	**DeepCNN**
No Attack	—	93.57 %	78.08 %	87.53 %
FGSM	epsilon=0.2	51.70 %	35.90 %	58.25 %
JSMA	gamma=0.2	57.57 %	72.56 %	61.50 %
Uni-FGSM	epsilon=0.2	4.10 %	4.39 %	8.97 %

It is worth noting that the selected applications were adapted from real-world IFTTT application scenarios, while their core logic was tailored to our dataset sensor types and implemented independently. We have also simulated the behavior of both a TAP's application execution engine and the privacy mediator for generating perturbations and filtering out fake actions.

We processed the 24-hour data from our dataset through the TAP in the normal state, without PTAP. During this period, the smart home dataset transmitted 28,186 data events to TAP for processing, triggering 252 application executions and the corresponding actions across the five applications. Subsequently, we processed the same data while incorporating JSMA-generated noise at a gamma value of 0.05. This added an additional 5,096 data events for TAP to process for the five applications, which led to 42 new application executions triggered by TAP, which in turn generated 42 fake actions. Given that we injected additional noise rather than modifying the real data, the privacy mediator was able to efficiently run IoT applications over the noisy data and filter out all resulting fake actions.

E. Computational and Network Overhead

The execution time of each adversarial attack provides insights into the latency introduced by PTAP. Our experiments revealed that on average the FGSM attack took 34.36 milliseconds per instance, while the JSMA attack took slightly longer, ranging from 327.16 to 1428.15 milliseconds per data sample due to varying attack parameters. This difference can primarily be attributed to the inherent complexity of the JSMA method compared to FGSM. Furthermore, the iterative loop in JSMA for larger gammas contributes to the extended duration, as detailed in Table III. For the universal adversarial perturbation (UAP), the time required to generate the perturbation varied significantly based on the chosen epsilon values, ranging from 42.06 seconds to 707.46 seconds. This longer initial computation time can be attributed to the fact that UAP generates a universal perturbation only once, which can then be applied to multiple data samples. Importantly, this computation can be performed prior to the real-time process. Once the perturbation is generated, UAP incurs no additional computation time during its application, which makes it more efficient than FGSM and JSMA in scenarios where the perturbation can be prepared in advance.

Another significant aspect of our evaluation is the number of injected events for each run. This metric offers insights into potential network overheads, as the count of events directly correlates with network communication and data storage requirements. On average, 48.11 events were injected per data sample in our FGSM attack, while the number of injected events per data sample in our JSMA attack ranged from 9.40 to 36.69 depending on the attack parameters. We interpreted each data sample as an aggregation of data events in a 60-second time window on the TAP server.

In comparison, the number of injected events in the JSMA attack is less than 5 percent of aggregated data events (considering an effective gamma of 0.05), while the FGSM attack strategy perturbs all numerical sensors with the minimum acceptable perturbation amount, determined by the epsilon parameter. Consequently, the number of injected events in the FGSM attack amounts to 24.5 percent of the aggregated data events. Hence, although the FGSM attack is faster than the JSMA attack, its network overhead is more significant due to the elevated count of perturbed events. This emphasizes the balance between computational efficiency and network resource utilization when dealing with adversarial attacks.

While these metrics emphasize the added latency and

Table III: Average computational overheads and number of injected events for FGSM, JSMA, and UAP attacks under varying parameters.

Attack	**FGSM (ϵ)**									
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Time (ms)	31.92	31.53	34.82	34.76	34.79	35.04	35.43	35.01	35.25	35.12
Average number of injected events (per sample)	48.11	48.11	48.11	48.11	48.11	48.11	48.11	48.11	48.11	48.11
Attack	**JSMA (γ)**									
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Time (ms)	327.16	540.10	720.67	875.85	1009.56	1090.99	1155.55	1331.1	1356.24	1428.15
Average number of injected events (per sample)	9.40	17.40	23.19	27.06	30.02	31.55	32.59	35.64	35.62	36.69
Attack	**UAP (ϵ)**									
	0.001	0.002	0.003	0.004	0.005	0.1	0.2	0.3	0.4	0.5
Time (sec)	707.46	136.51	61.85	61.10	59.63	48.39	50.29	50.77	42.19	42.06
Average number of injected events (per sample)	160.6	169.6	163.9	166.62	159.7	153	164.4	162.1	163.3	135.7

communication overheads, it is essential to understand them within the broader context of IoT applications where slight delays are tolerable. Notably, commercial TAPs like IFTTT inherently exhibit delays. As highlighted in [35], some triggers provided by IFTTT are polling-based, operating at a frequency of once every 15 minutes. This feature indicates that IFTTT is not optimized for precise real-time service. In this light, the marginal delay of up to 1.42 seconds introduced by PTAP is deemed reasonable in real-world scenarios.

VI. Discussion

In this section, we discuss various aspects and limitations of PTAP, and identify potential avenues for future research.

A. Individual Data Points and Applicability

Sensitivity of single data points. The current focus of PTAP is on inhibiting sensitive inferences. However, there may be situations where individual data points themselves hold a degree of sensitivity. For example, consider an IFTTT application that activates a plug upon the user's arrival home [47]. This application necessitates the use of location data, which could be viewed as sensitive information. As a default, we proceed under the presumption that the user has given consent to the sharing of this kind of data, considering its pertinence to the application's functionality. In future work, our aim is to include the sensitivity of single data points into our privacy considerations using anonymization techniques.

Handling external action services. PTAP uses a local copy of the trigger-action rules to identify fake actions arising from fake data. However, in certain scenarios, the action service of an IoT application could be an external cloud-based service, outside the user's control. For instance, consider an IFTTT application that sends a notification email when power consumption exceeds a threshold [61]. In this case, the privacy mediator has no access to the generated fake action resulting from the perturbation, and therefore the action cannot be filtered. This means that the user would receive fake emails if fake data were generated. To maintain the utility of services that cannot be replicated locally for filtering, PTAP currently prevents the perturbation of data that is connected to such actions.

Exploring Further Applicability. Our results are based on the Orange4Home dataset. While we expect our findings to apply equally to similar setups and sensor data types, further experiments are necessary to validate their applicability across a broader range of scenarios and platforms.

B. Advanced Attacks

Adversarial adaptive attacker. In case the attacker, a malicious TAP, is aware of our privacy-preserving mechanism, it might adapt its strategies to circumvent PTAP. In such a scenario, the attacker might implement adversarial training to counteract our adversarial example generation. Here, the TAP continuously updates its models based on the perturbed data to improve its ability to classify user activities despite the noise. In real-world scenarios, this tactic is unlikely. Adversarial training of activity recognizer in the TAP requires knowledge about the real activity class labels. These are typically inaccessible to the TAP, as only the user knows the real activity class labels. For a specific case where labeled data is available for training the attacker classifier, the TAP can conduct adversarial training to reduce PTAP's effectiveness significantly. Nevertheless, previous work [52] presents ideas for robust adversarial examples tailored to specific users, which can offer resistance against adversarial training by clouding a specific user's data. Our future works will aim to improve PTAP's robustness against adversarial training from an adaptive attacker.

Addressing semantic relations among inferences. In some scenarios, extracted inferences might have a semantic relation to each other, which could potentially leak information allowing the attacker to discern that a fake generated inference is not real based on possible implications of related inferences. For example, if a user aims to hide their activity classes from a malicious TAP classifier, a sophisticated TAP might infer the probable relation between occurred and upcoming activities, and deduce the presence of fake activities, thus posing a challenge for PTAP. Currently, PTAP does not take into account any relations between activities during adversarial example generation, which implies an untargeted adversarial generation approach. Nonetheless, by integrating a simple upcoming activity class prediction, PTAP could leverage a targeted adversarial example generation strategy that offers robustness against possible inference leaks between fake generated activities. This is a point we aim to explore in future developments.

Targeting privacy mediator. The privacy mediator in PTAP maintains local copies of a user's IoT applications and has access to their data events, therefore posing privacy risks in case of compromise. We argue that the risk is drastically reduced compared to the attacker model of a compromised or untrusted TAP. In fact, thanks to PTAP, an attacker would now need to compromise every user's privacy mediator across different networks to gain similar access to compromising the TAP, which is more difficult.

C. Privacy

Privacy guarantees. While practical experiments have demonstrated the significant impact of PTAP in reducing the accuracy of malicious TAP classifiers, there is currently no absolute guarantee for privacy protection. This issue is inherently associated with the nature of adversarial example attacks and their transferability feature, which do not ensure absolute privacy preservation and are practically effective. We aim to develop theoretically-backed privacy guarantees to enhance the dependability of our privacy-preserving solution.

VII. Related Work

Privacy on TAPs. Several strategies have been proposed in recent literature to address the privacy concerns of TAPs. These strategies can be broadly classified into approaches based on data minimization and data encryption.

Data minimization approaches aim to enforce the principles of least privilege and need-to-know. F&F [64] proposes minimization of data transfer within a threat model that involves a trustworthy SmartThings platform communicating with a potentially malicious IFTTT platform. Similarly, minTAP [14] employs a language-based data minimization strategy that discloses to the TAP only the necessary user data attributes pertaining to the execution of an IoT application. PFirewall [16], on the other hand, filters user data based on automation-dependent and user-specified data-minimization policies. SandTrap [2] proposes language-based sandboxing to isolate and mediate communication between different IoT applications under the threat model of a trusted TAP. Other approaches track the flow of information in trusted TAPs with the goal of identifying malicious or buggy IoT applications [6, 11, 12, 21].

In contrast to data minimization approaches, PTAP does not merely limit data sharing. Instead, it uses a novel strategy of injecting perturbations into data, misleading classifiers while preserving user utility.

Data encryption approaches aim to enforce confidentiality of each data point. OTAP [17] leverages encryption algorithms at the expense of imposing constraints on users, limiting them to IoT applications devoid of computation, which might be restrictive for many users. Exploring a different route, eTAP [15] executes IoT applications without accessing user's data in plain text, using a garbled circuit protocol. In contrast to these encryption-centric methods, PTAP considers user privacy from the perspective of obscuring sensitive inferences, rather than encrypting all data points. Consequently, PTAP manipulates data without the need for encryption. This approach allows PTAP to seamlessly integrate into the existing smart home TAP architecture without necessitating alterations to the TAP cloud, IoT hub, user client, or the IoT devices' communication—alterations that would be necessary to implement encryption-based approaches such as OTAP and eTAP.

Data generation has been used in other domains to protect user privacy. For example, in the context of Android apps, Mockdroid [7] generates fake data for a predefined set of properties that are prevalent in mobile applications, using custom-tailored algorithms. PTAP is general and can generate data for any sensor domain using configurable algorithms. Since the fake data can affect IoT devices through trigger-action platforms, PTAP additionally requires a utility-preserving mechanism to filter out fake actions that would otherwise be triggered in a smart-home environment.

Privacy-preserving data analysis. Extensive research has addressed concerns surrounding the significant amount of sensitive information that can be derived from the data collected by mobile, IoT, and wearable devices. Replacement AutoEncoder (RAE) [33] introduces a framework to manage access to time-series data with the intention of shielding temporal inferences. RAE learns to transform the discriminative features of data that correspond to sensitive inferences into features that are more often observed in non-sensitive inferences, hence safeguarding user privacy. In another work, Guardian Estimator Neutralizer (GEN) [32] aims to protect concurrent inferences. Inspired by GANs, GEN establishes a game between a data transformer model (guardian) and an information extractor model (estimator) to achieve an efficient data transformation that ensures a balance between utility and privacy. Also, the rapidly growing field of machine learning has prompted considerable interest in applying differential privacy to deep learning to balance privacy needs with large, representative datasets. Abadi et al. [1] propose a novel method for differential privacy, focusing on training deep neural networks. In contrast to these methods, our proposed approach, PTAP, takes into account the unique context of TAPs and aims to preserve the functionality of IoT applications, thereby enhancing user privacy without compromising utility.

Adversarial example for privacy. Adversarial examples have been utilized as an effective strategy to preserve privacy in various contexts. The generated perturbations attempt to misdirect machine learning models while maintaining the original data's utility.

Chen et al. [13] propose a voice de-identification system that has been developed to mitigate the privacy-utility trade-off often faced by users. Their system uses convolutional adversarial examples to hide user identity from Automatic Speaker Identification (ASI). In a different context, Sadeghzadeh et al. [52] propose a defense against website fingerprinting attacks using adversarial deep learning approaches to preserve the privacy of user browsing activities. In another work, Liu et al. [30] propose a novel Stealth algorithm designed to blind automatic deep neural network (DNN) detectors to the

presence of objects in an image, without compromising the visual quality perceptible to human eyes by generating a type of adversarial example that effectively renders the objects "invisible" to the detector. Despite the breadth of applications where adversarial examples have shown promise in enhancing privacy, their potential in the context of TAPs was unexplored. To the best of our knowledge, PTAP is the first work that employs adversarial examples to address privacy concerns in the context of TAPs.

VIII. Conclusion

In this paper, we have presented a privacy preservation framework for smart homes leveraging adversarial examples. These adversarial examples aim to disrupt TAP's classifier accuracy with controlled sensor data perturbations, thereby safeguarding smart home privacy. Our work has shown that PTAP significantly reduces the accuracy of a malicious classifier, from 91 % to 6 %. Our filter confirms PTAP's role in upholding IoT application integrity, demonstrating its efficacy in smart-home privacy preservation.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and feedback. Additionally, we extend our gratitude to Mohammad Ahmadpanah, Amir Mahdi Sadeghzadeh, and Behrad Tajali for their helpful feedback on this work. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and the Swedish Research Council (VR).

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, https://dl.acm.org/doi/pdf/10.1145/2976749.2978318, 2016. ACM SIGSAC.

[2] Mohammad M. Ahmadpanah, Daniel Hedin, Musard Balliu, Lars Eric Olsson, and Andrei Sabelfeld. SandTrap: Securing JavaScript-driven Trigger-Action Platforms. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2899–2916. USENIX Association, 2021.

[3] Noah Apthorpe, Dillon Reisman, and Nick Feamster. Closing the blinds: Four strategies for protecting smart home privacy from network observers, 2017. arXiv:1705.06809.

[4] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic, 2017. arXiv:1708.05044.

[5] Musard Balliu, Iulia Bastys, and Andrei Sabelfeld. Securing IoT Apps. *IEEE Security & Privacy*, 17(5):22–29, 2019.

[6] Iulia Bastys, Musard Balliu, and Andrei Sabelfeld. If this then what? Controlling flows in IoT apps. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1102–1119, https://dl.acm.org/doi/pdf/10.1145/3243734.3243841, 2018. ACM SIGSAC.

[7] Alastair R. Beresford, Andrew Rice, Nick Skehin, and Ripduman Sohan. Mockdroid: Trading Privacy for Application Functionality on Smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, pages 49–54. ACM, March 2011.

[8] Zeki Bilgin, Emrah Tomur, Mehmet Akif Ersoy, and Elif Ustundag Soykan. Statistical appliance inference in the smart grid by machine learning. In *2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops)*, pages 1–7, Istanbul, Turkey, 2019. IEEE, IEEE.

[9] Damien Bouchabou, Sao Mai Nguyen, Christophe Lohr, Benoit LeDuc, and Ioannis Kanellos. A survey of human activity recognition in smart homes based on IoT sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. *Sensors*, 21(18):6037, 2021.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, San Jose, CA, USA, 2017. IEEE, IEEE.

[11] Z Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A Selcuk Uluagac. Sensitive information tracking in commodity IoT. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1687–1704, Baltimore, USA, 2018. Usenix.

[12] Z. Berkay Celik, Gang Tan, and Patrick D. McDaniel and. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *NDSS*, 2019.

[13] Meng Chen, Li Lu, Jiadi Yu, Yingying Chen, Zhongjie Ba, Feng Lin, and Kui Ren. Privacy-Utility Balanced Voice De-Identification Using Adversarial Examples. https://arxiv.org/pdf/2211.05446, 2022.

[14] Yunang Chen, Mohannad Alhanahnah, Andrei Sabelfeld, Rahul Chatterjee, and Earlence Fernandes. Practical Data Access Minimization in Trigger-Action Platforms. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2929–2945, Boston, MA, USA, 2022. USENIX.

[15] Yunang Chen, Amrita Roy Chowdhury, Ruizhe Wang, Andrei Sabelfeld, Rahul Chatterjee, and Earlence Fernandes. Data privacy in trigger-action systems. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 501–518, San Francisco, CA, 2021. IEEE.

[16] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Lannan Luo. PFirewall: Semantics-Aware Customizable Data Flow Control for Smart Home Privacy Protection. In *28th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2021. The Internet Society, NDSS.

[17] Yu-Hsi Chiang, Hsu-Chun Hsiao, Chia-Mu Yu, and Tiffany Hyun-Jin Kim. On the privacy risks of compromised trigger-action platforms. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25*, pages 251–271, Guildford, UK, 2020. Springer, Springer.

[18] Julien Cumin, Grégoire Lefebvre, Fano Ramparany, and James L Crowley. A dataset of routine daily activities in an instrumented home. In *Ubiquitous Computing and Ambient Intelligence: 11th International Conference, UCAmI 2017, Philadelphia, PA, USA, November 7–10, 2017, Proceedings*, pages 413–425, Philadelphia, PA, USA, 2017. Springer, Springer.

[19] ecobee. Humidity Control. IFTTT Applet: Humidity Control, 2023. Accessed: 2023-05-04.

[20] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 636–654, San Jose, CA, USA, 2016. IEEE Computer Society.

[21] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks. In *USENIX Security*, 2016.

[22] Furkan Goksel, Muslum Ozgur Ozmen, Michael Reeves, Basavesh Shivakumar, and Z Berkay Celik. On the safety implications of misordered events and commands in IoT systems. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 235–241, San Francisco, CA, USA, 2021. IEEE, IEEE.

[23] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint, 2014. arXiv:1412.6572.

[24] Rebeen Ali Hamad, Longzhi Yang, Wai Lok Woo, and Bo Wei. Joint learning of temporal models to handle imbalanced data for human activity recognition. *Applied Sciences*, 10(15):5293, 2020.

[25] Ian Goodfellow and Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[26] Harris Interactive. Consumer Internet of Things security labeling survey research findings. Report for UK Gov. DDCMS, 2019.

[27] Dongwan Kang. Samsung Developer Conference 2023 (SDC23). YouTube: SDC23 Keynote, 2023. Accessed: 2023-05-04.

[28] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, Boca Raton, FL, 2018.

[29] Beatrice Li, Arash Tavakoli, and Arsalan Heydarian. Occupant privacy perception, awareness, and preferences in smart office environments. *Scientific Reports*, 13(1):4073, 2023.

[30] Yujia Liu, Weiming Zhang, and Nenghai Yu. Protecting privacy in shared photos via adversarial examples based stealth. *Security and Communication Networks*, 2017, 2017.

[31] Anindya Maiti and Murtuza Jadliwala. Light ears: Information leakage via smart lights. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–27, 2019.

[32] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, pages 1–6, Porto Portugal, 2018. ACM.

[33] Mohammad Malekzadeh, Richard G Clegg, and Hamed Haddadi. Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis. arXiv preprint, 2017. arXiv:1710.06564.

[34] Javier Medina-Quero, Shuai Zhang, Chris Nugent, and Macarena Espinilla. Ensemble classifier of long short-term memory with fuzzy temporal windows on binary sensors for activity recognition. *Expert Systemfus with Applications*, 114:441–453, 2018.

[35] Xianghang Mi, Feng Qian, Ying Zhang, and XiaoFeng Wang. An empirical characterization of IFTTT: ecosystem, usage, and performance. In *Proceedings of the 2017 Internet Measurement Conference*, pages 398–404, New York, USA, 2017. ACM.

[36] Alaeddine Mihoub. A deep learning-based framework for human activity recognition in smart homes. *Mobile Information Systems*, 2021:1–11, 2021.

[37] Roghayeh Mojarad, Ferhat Attal, Abdelghani Chibani, and Yacine Amirat. A context-aware hybrid framework for human behavior analysis. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 460–465. IEEE, 2020.

[38] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66, Seattle, USA, 2010. ACM.

[39] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, Honolulu, HI, USA, 2017. IEEE.

[40] Pardis Emami Naeini, Sruti Bhagavatula, Hana Habib, Martin Degeling, Lujo Bauer, Lorrie Faith Cranor, and Norman Sadeh. Privacy expectations and preferences in an IoT world. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 399–412, Santa Clara, CA, USA, 2017. USENIX Association Santa Clara, USENIX.

[41] OpenHAB. open Home Automation Bus. OpenHAB Website, 2019. Accessed: 2023-05-04.

[42] Edewede Oriwoh and Marc Conrad. Presence detection from smart home motion sensor datasets: a model. In *XIV Mediterranean Conference on Medical and Biological*

Engineering and Computing 2016: MEDICON 2016, pages 1249–1255, Paphos, Cyprus, 2016. Springer.

[43] Orro. When you unlock your door, turn on the lights. IFTTT Applet: Unlock Door, Lights On, 2023. Accessed: 2023-05-04.

[44] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint, 2016. arXiv:1605.07277.

[45] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, Abu Dhabi, United Arab Emirates, 2017. ACM.

[46] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387, Germany, 2016.

[47] Hive Active Plug. Turn your plug on when you arrive home. IFTTT Applet: Turn Plug On at Home Arrival, 2023. Accessed: 2023-05-04.

[48] Kevin Purdy. Appliance makers sad that 50% of customers won't connect smart appliances. Ars Technica Article: Smart Appliance Connectivity, 2023. Accessed: 2023-05-04.

[49] P. Rathore, A. Basak, S. H. Nistala, and V. Runkana. Untargeted, targeted and universal adversarial attacks and defenses on time series. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, July 2020.

[50] Ring. Turn on your Hue light when Ring detects motion at your door. IFTTT Applet: Hue Light on Motion Detection, 2023. Accessed: 2023-05-04.

[51] Rixin Xu and Qiang Zeng and Liehuang Zhu and Haotian Chi and Xiaojiang Du and Mohsen Guizani. Privacy leakage in smart homes and its mitigation: IFTTT as a case study. *IEEE Access*, 7:63457–63471, 2019.

[52] Amir Mahdi Sadeghzadeh, Behrad Tajali, and Rasool Jalili. AWA: Adversarial website adaptation. *IEEE Transactions on Information Forensics and Security*, 16:3109–3122, 2021.

[53] Sandy Schoettler, Andrew Thompson, Rakshith Gopalakrishna, and Trinabh Gupta. Walnut: A low-trust trigger-action platform. arXiv preprint, 2020. arXiv:2009.12447.

[54] Smappee. Turn off stuff when my consumption is too high. IFTTT Applet: Turn off on High Consumption, 2023. Accessed: 2023-06-04.

[55] SmartThings. SmartThings Privacy Notice. SmartThings Privacy Policy, 2022. Accessed: 2023-06-04.

[56] SmartThings Blog. Developers Are Building the Next Era of Smart Homes and Buildings with new SmartThings APIs. SmartThings Blog: New SmartThings APIs, 2023. Accessed: 2023-10-05.

[57] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint, 2013. arXiv:1312.6199.

[58] Google Nest Thermostat. Turn on Midea AC when your indoor temperature rises too high. IFTTT Applet: Turn on Midea AC, 2023. Accessed: 2023-05-04.

[59] Daphne Townsend, Frank Knoefel, and Rafik Goubran. Privacy versus autonomy: a tradeoff model for smart home monitoring technologies. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4749–4752, Boston, Massachusetts, USA, 2011. IEEE, IEEE.

[60] Timotheus Leonhard Martinus van Kasteren et al. *Activity recognition for health monitoring elderly using temporal probabilistic models*. PhD thesis, ASCI, 2011.

[61] Vimar VIEW. Single phase total power consumption threshold - Email notification. IFTTT Applet: Power Consumption Threshold, 2023. Accessed: 2023-05-08.

[62] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern recognition letters*, 119:3–11, 2019.

[63] Charlie Wilson, Lina Stankovic, Vladimir Stankovic, Jing Liao, Michael Coleman, Richard Hauxwell-Baldwin, Tom Kane, Steven Firth, and Tarek Hassan. Identifying the time profile of everyday activities in the home using smart meter data, 2015. Accessed: May 11, 2023.

[64] Rixin Xu, Qiang Zeng, Liehuang Zhu, Haotian Chi, Xiaojiang Du, and Mohsen Guizani. Privacy leakage in smart homes and its mitigation: IFTTT as a case study. *IEEE Access*, 7:63457–63471, 2019.

[65] Yi Xu, Jan-Michael Frahm, and Fabian Monrose. Watching the Watchers: Automatically Inferring TV Content from Outdoor Light Emissions. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 418–428, Scottsdale, Arizona, USA, 2014. ACM.

[66] Longqi Yang, Kevin Ting, and Mani B Srivastava. Inferring occupancy from opportunistically available sensor data. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 60–68, Budapest, Hungary, 2014. IEEE, IEEE.

[67] Ibrahim Yilmaz and Ambareen Siraj. Avoiding occupancy detection from smart meter using adversarial machine learning. *IEEE Access*, 9:35411–35430, 2021.

[68] Yu-Hsi Chiang and Hsu-Chun Yu and Chia-Mu Yu and Tiffany Hyun-Jin Kim. On the privacy risks of compromised trigger-action platforms. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25*, pages 251–271, Guildford, UK, 2020. Springer.

[69] Xu Zheng, Zhipeng Cai, and Yingshu Li. Data linkage in smart IoT systems: a consideration from a privacy perspective. *IEEE Commun. Mag.*, 56(9):55–61, 2018.