Safeguarding User-Centric Privacy in Smart Homes

KEYANG YU, Computer Science, Colorado School of Mines, Golden, United States
QI LI, Computer Science, Colorado School of Mines, Golden, United States
DONG CHEN, Computer Science, Colorado School of Mines, Golden, United States
LITING HU, University of California Santa Cruz, Santa Cruz, United States

Internet of Things (IoT) devices have been increasingly deployed in smart homes to automatically monitor and control their environments. Unfortunately, extensive recent research has shown that on-path external adversaries can infer and further fingerprint people's sensitive private information by analyzing IoT network traffic traces. In addition, most recent approaches that aim to defend against these malicious IoT traffic analytics cannot adequately protect user privacy with reasonable traffic overhead. In particular, these approaches often did not consider practical traffic reshaping limitations, user daily routine permitting, and user privacy protection preference in their design. To address these issues, we design a new low-cost, open source user-centric defense system—PrivacyGuard—that enables people to regain the privacy leakage control of their IoT devices while still permitting sophisticated IoT data analytics that is necessary for smart home automation. In essence, our approach employs intelligent deep convolutional generative adversarial network assisted IoT device traffic signature learning, long short-term memory based artificial traffic signature injection, and partial traffic reshaping to obfuscate private information that can be observed in IoT device traffic traces. We evaluate PrivacyGuard using IoT network traffic traces of 31 IoT devices from five smart homes and buildings. We find that PrivacyGuard can effectively prevent a wide range of state-of-the-art adversarial machine learning and deep learning based user in-home activity inference and fingerprinting attacks and help users achieve the balance between their IoT data utility and privacy preserving.

CCS Concepts: • **Information systems** → Data management systems; • **Computer systems organization** → **Sensor networks**; • **Security and privacy** → **Vulnerability management**; **Information flow control**; • **Computing methodologies** → **Model development and analysis**; *Machine learning algorithms;*

Additional Key Words and Phrases: Security and privacy, data analytics, IoT sensors and devices, modeling and analysis

ACM Reference Format:
Keyang Yu, Qi Li, Dong Chen, and Liting Hu. 2024. Safeguarding User-Centric Privacy in Smart Homes. *ACM Trans. Internet Technol.* 24, 4, Article 23 (November 2024), 33 pages. https://doi.org/10.1145/3701726

Corresponding author: Dong Chen, Colorado School of Mines.
This research was supported by NSF grant 2238701.
Authors' Contact Information: Keyang Yu, Computer Science, Colorado School of Mines, Golden, Colorado, United States; e-mail: yukeyang@mines.edu; Qi Li, Computer Science, Colorado School of Mines, Golden, Colorado, United States; e-mail: liqi@mines.edu; Dong Chen, Computer Science, Colorado School of Mines, Golden, Colorado, United States; e-mail: dongchen@mines.edu; Liting Hu, University of California Santa Cruz, Santa Cruz, California, United States; e-mail: liting@ucsc.edu.

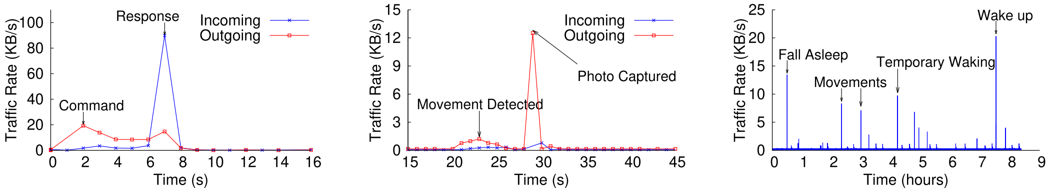
ACM 1533-5399/2024/11-ART23
https://doi.org/10.1145/3701726

1 Introduction

People are increasingly deploying the **Internet of Things (IoT)** devices in smart homes and buildings to monitor and control their environments. The total installed base of IoT devices is projected to reach 75.44 billion worldwide by 2025, a fivefold increase over 10 years [50]. The traffic data generated by IoT devices is recorded by **Internet Service Providers (ISPs)** to maintain customer services, such as generating monthly bills, personalizing the data plan, and detecting network outages. "Any service that provides Internet access can obviously see what resources users are accessing. And even with encryption, traffic patterns provide some information about activity" [35]. Verizon uses "supercookies" to track Internet user activity, and AT&T charges customers an extra $29 per month to avoid "the collection and monetization of their browsing history for targeted ads," Mozilla told Congress [36]. ISPs like AT&T, Comcast, Time Warner, Sprint, and Verizon are selling personal network traffic data without prior user consent to "enhance" user experience [11]. Specially, the most recent IoT privacy survey [32] shows that 72 out of 81 popular IoT devices are sharing data with third parties (e.g., Amazon, Akamai) completely unrelated to original manufacturers and far beyond basic necessary device configuration, including voice speakers, smart TVs, and streaming dongles. People are losing access to traffic data and sharing control of their IoT devices at home.

In parallel, significant recent research [9, 10, 12, 13, 18, 21, 26, 28, 38, 39, 47, 56, 57] has shown that it is surprisingly easy to launch attacks to infer the types of IoT devices and user activities from IoT traffic data, since IoT device types and their embedded user activities are highly correlated with basic statistical metrics derived from time series data, such as mean, variance, and range. Thus, IoT devices have significant privacy threats in their network traffic data. An important example of simple and private information that IoT traffic data may leak is *occupancy*—whether or not someone is at home and when [31]. The network traffic rate (in kilobytes per second) of three IoT devices from a single apartment is reported in Figure 1. The traffic rate trace signals the occupancy status and related activities in this home. The most recent research [29] also demonstrated that a passive Amazon Alexa attacker can fingerprint user voice commands and compromise the user privacy of millions of U.S. consumers. In addition, these IoT traffic data may also indirectly reveal privacy information that might be interesting for insurance companies, marketers, or the government. For instance, significant traffic spikes at mealtimes may indicate that users are regularly having meals at home. As another example a consistent amount of TV network traffic on Saturday night from 7 pm to 9 pm may indicate that the residents are watching NBA games every weekend. In addition, a lack of significant traffic may show that occupants are out of town for vacation. Intuitively, user interaction with IoT devices, such as talking to voice assistants, opening/closing doors, and watching TVs, lends itself to straightforward attacks that detect changes in these metrics and associates them with changes in user activities.

Most recent research [9, 12, 16, 37, 54–57] proposes traffic reshaping based prevention techniques to thwart privacy attacks on IoT traffic rate traces. Unfortunately, these approaches did not significantly consider at least one of the following facts: (1) The artificial traffic "spikes" that are injected to hide user privacy should not conflict with the real user behavior; (2) many IoT devices have bidirectional network traffic flows that should be reshaped concurrently; and (3) reshaping operations have practical limitations, such as network bandwidth and maximum injection rate. And these may still allow adversaries to infer user in-home sensitive private information. In addition, the native flattening algorithms broadly employed by many approaches resulted in three to four times additional traffic overhead. Thus, new low-cost and effective techniques are necessary. To address these issues, we propose a new low-cost, open source user "tunable" defense system—PrivacyGuard—that enables users to significantly reduce the private information leaked through IoT device network traffic while still permitting sophisticated traffic analytics that is necessary to use IoT devices. In doing so, we make the following contributions.



(a) Amazon Alexa (b) Dropcam Wireless Camera (c) Withings Sleep Sensor

Fig. 1. When occupied, the IoT device traffic rate typically becomes larger and more variable due to user interactions.

User Privacy Leakage Identification. We explore and highlight the privacy leakage of user in-home activities from IoT network traffic rate traces. We discuss the fundamental privacy concerns that govern the network traffic rate over time for popular IoT devices. In doing so, we review, implement, and benchmark a wide range of sophisticated user activity inference attack models using **Machine Learning (ML)** or **Deep Learning (DL)** approaches, including k-nearest neighbors, **Hidden Markov Models (HMMs)**, **Support Vector Machines (SVMs)**, **Convolutional Neural Network (CNNs)**, and an ensemble approach—AdaBoost [23].

PrivacyGuard Design. We present the design of PrivacyGuard, which enables users to regain the privacy leakage control of their IoT devices to significantly reduce the private information leaked through IoT device network traffic. In essence, PrivacyGuard employs intelligent **Deep Convolutional Generative Adversarial Network (DCGAN)**-based traffic signature learning, **Long Short-Term Memory (LSTM)**-assisted artificial traffic signature injection, and partial traffic reshaping to obfuscate user privacy. We also design optimization techniques to further reduce PrivacyGuard's traffic overhead.

Implementation and Evaluation. We implement PrivacyGuard, both simulator and prototype, in `python` using the widely used open source frameworks. We evaluate PrivacyGuard using traffic rate traces of 31 different IoT devices from five smart homes. The results have shown that PrivacyGuard can effectively prevent a wide range of state-of-the-art ML/DL-based user activity inference attacks.

Releasing Datasets and Code. Our new approaches to analyze IoT network traffic rate traces and prevent user sensitive information from leakage in these traces using ML/DL-based traffic reshaping techniques are quite general, and can be applied to address similar security and privacy problems in other data analytics research domains, such as smart transportation system, smart grid, and medical e-health systems. We release the source code, datasets, and attack models of PrivacyGuard to IoT research communities on our website [3].

2 Background and Related Work

2.1 Privacy Threat Model

As shown in Figure 2, we are broadly concerned with the ability of ISPs, on-path network observers, and third parties to infer user in-home activities from smart home network traffic rate metadata. The network traffic rate metadata, including inbound/outbound traffic rates, network protocols, source, and destination IPs, package sizes, and so forth are accessible to many on-path entities. And these potential adversaries may be incentives to infer user activities in smart homes where users do not want to share this privacy-sensitive information with them. We assume that external adversaries can use any data analytics techniques, such as data mining, ML/DL, inference, or other statistical methods, to infer certain types of the observed pattern information in the recorded traffic

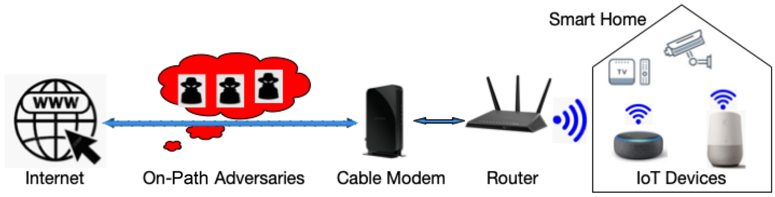
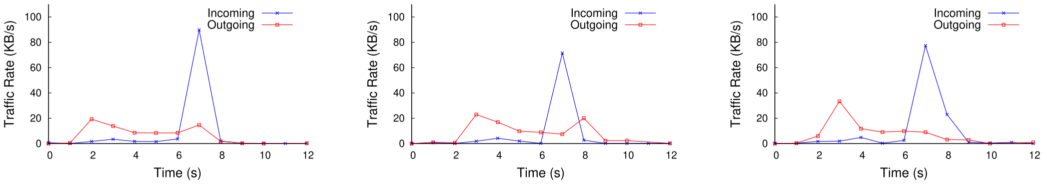
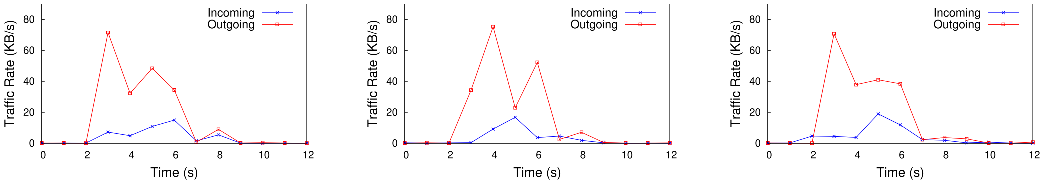


Fig. 2. Overview of our privacy threat model in smart homes and smart buildings.



(a) User 1 (b) User 2 (c) User 3

Fig. 3. Traffic rate signatures using Amazon Alexa generated by querying the weather condition from three different smart home users.



(a) User 1 (b) User 2 (c) User 3

Fig. 4. Fingerprinting traffic rate signatures using Google Home to control the light of three different smart home users.

traces. Thus, inferring user activities in these smart homes is considered as an opposition to users' privacy preferences.

In particular, we are concerned with four different types of privacy attacks. The first type is *learning occupancy from the data.* This includes whether a home is occupied and when. The second is *learning user in-home activities from the traffic data.* User activities may include when users come and go, when they perform their daily activities, such as going to bed, waking up, watching TV, listening to music, and playing online games, as well as more complex questions, such as whether a household has a baby, and whether they go on vacation on weekends. The third type is *learning network traffic pattern information from the data.* This includes whether a particular IoT device (e.g., Voice Assistant) is present in a home, what model of an IoT device is present, and how much traffic the home consumes on it every month. The fourth is *fingerprinting voice command on user interactive IoT devices.* As shown in Figures 3 and 4, this may include voice command fingerprinting on smart home speakers, such as Alexa and Google Nest Home. Inferring this kind of detailed voice command may further expose more serious user private information.

Attack Scenario #1. To infer the type of IoT devices and user activities at a certain home, an external Internet on-path adversary intends to acquire real-time network traffic rates and leverages ML/DL-based statistical learning and data mining approaches to analyze these data to identify IoT

Table 1. Correlation and Traffic Overhead Comparison of Six Major Traffic Modification Approaches

	PCC	SRCC	Security (ϵ)	Additional Overhead
Pure Traffic Injection	0.748	0.83	87.15%	97%
Hybrid Traffic Reshaping	0.462	0.711	72.6%	103.7%
Random Traffic Padding	0.582	0.686	54.33%	165.9%
Tor [12]	0.805	0.712	77.5%	25%
RepEL [10]	0.361	0.525	33%	100%
Tamaraw [12]	0.292	0.473	3.4%	199%

devices. Then, the external attacker may launch cyberattacks on a specific IoT device when user activities permit.

Attack Scenario #2. An external adversary from ISPs, IoT device manufacturers, or third parties is actively monitoring the IoT traffic traces and then uses data analytics approaches to learn the indirect user privacy information that might be interesting for insurance companies, marketers, or the government.

Attack Scenario #3. An external on-path adversary is actively monitoring the IoT traffic traces from a smart home and then launches inference attacks to fingerprint user voice commands that might be interesting for insurance companies, marketers, or the government.

In addition, we assume that our smart home users would like to trust in Amazon AWS (EC2) or Google Cloud services to protect their in-home user privacy information. Note that evaluating the effectiveness of establishing trust relationship between end users and their cloud servers is outside the scope of this article.

2.2 Related Work

We outline the design alternatives to preserve smart home user privacy using the **Pure Traffic Injection (PTI)** approach, the **Hybrid Traffic Reshaping (HTR)** approach, and the **Random Traffic Padding (RTP)** approach. In doing so, we review a wide range of the most recent sophisticated traffic reshaping based prevention techniques [9, 10, 12, 13, 21, 25, 28, 39, 44–47, 56, 57] to thwart privacy attacks on IoT traffic rate traces.

To understand the performance of the preceding existing approaches, we implemented three different traffic reshaping approaches. Table 1 quantifies the effectiveness of the three approaches and an additional three recent approaches by showing the **Pearson Correlation Coefficient (PCC)** and **Spearman's Rank Correlation Coefficient (SRCC)**. The PCC [40] is a measure of the linear correlation between original and modified traffic. It has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation. The SRCC [49] assesses monotonic relationships between original traffic and modified traffic. If there are no repeated data values, a perfect SRCC of +1.0 or −1.0 occurs when each of the variables is a perfect monotone function of the other. Although recent approaches have been proposed to mitigate the privacy leakage issue, the modified traffic rate traces after applying these prior approaches may still have a very high linear and monotonic correlation with the original traffic rate traces. We use PCC and SRCC to quantify the effectiveness of the prior approaches on masking user private information. We use ϵ-security [39] to describe the probability that a traffic reshaping approach fails to prevent smart home users from an external adversary's user activity inferring.

Pure Traffic Injection. Prior work [12, 24, 33, 39] proposed defense approaches to inject "fake" traffic patterns to conceal genuine user network traffic patterns. As shown in Table 1, the general implementation yields additional overhead as 97%. In particular, Park et al. [39] found that traffic data encryption cannot prevent privacy invasions exploiting traffic pattern analysis and statistical inference. They first developed empirical models to statistically learn user behaviors using the transition status of wireless sensors. Then, cloaking network traffic patterns are injected to obscure genuine traffic patterns. Cai et al. [12] presented a defense against Tor website fingerprinting that can reshape traffic rate traces by controlling the size of the parameter to pad packets. Hafeez et al. [24] developed a traffic morphing approach to protect against traffic analytics attack. To obfuscate background traffic, they injected traffic at a constant rate, incorporating dummy traffic to simulate device events and blend genuine and fake traffic seamlessly. Another recent work [60] focuses on ON/OFF traffic shaping to mask user information in network traffic package and streaming data. However, these approaches did not completely hide the genuine traffic patterns, in particular, during higher and lower traffic periods. This may still allow adversaries to distinguish "fake" traffic patterns from genuine traffic patterns to infer user activities.

Hybrid Traffic Reshaping. Prior work [10, 13] presented hybrid reshaping techniques to prevent user privacy leakage in the aggregated network traffic data. These approaches are aiming at combining partial demand flattening and random artificial signature injection to obscure user privacy in the recorded data, and leveraging activity-aware optimizations to reduce their reshaping overhead. As shown in Table 1, the general implementation yields additional overhead as 103.7%. Chen et al. [13] proposed to learn the "noise" injection rate using empirical statistical analytics (e.g., probability mass function) of smart home device events. Similarly, Bovornkeeratiroj et al. [10] proposed RepEL, which employed an edge gateway to partially flatten loads and randomly replay loads to hide private user occupancy information. A differential privacy (d^*-Privacy [58])-based defense on voice speaker traffic privacy leakage was introduced by Wang et al. [53]. The defense design and evaluation are mainly for voice speaker traffic analysis. Uddin et al. [51] presented a software defined network motivated framework to protect user privacy from local or internal attackers. The proposed defense is focusing on packet level traffic padding and packet delaying. Alshehri et al. [8] proposed an STTA (Signature-based Tunneled Traffic Analysis) attack [8] that can be effective even on tunneled traffic. They designed a defense mechanism based on adding uniform random noise to protect against traffic analysis attack without introducing too much overhead. Xiong et al. [59] proposed a local differential privacy motivated defense mechanism to obfuscate IoT device packets prior to transmission in the presence of a local eavesdropper. Pinheiro et al. [42] designed an adaptive packet padding approach for smart home networks. Their traffic reshaping mechanisms particularly consider smart home network bandwidth and utilization to dynamically adjust their padding speed and volume. Shmatikov and Wang [47] proposed adaptive padding algorithms to leverage the intermediate mixes to inject dummy packets into statistically unlikely gaps in the packet flow to destroying timing "fingerprints" application traffic by enforcing inter-package intervals to match pre-defined probability mass functions. Wang et al. [57] designed a traffic padding algorithm that uses matched package schedules to prevent adversaries from pairing incoming and outgoing traffic flows. Significant work [25, 44–46] proposed to model user in-home activities using Markov chain based approaches. However, due to the empirical modeling of IoT device events and the nature of random traffic signature injection, these approaches may still allow smart attackers to identify the randomly injected "fake" signatures and thus infer the genuine user private information.

Random Traffic Padding. Recent work [9, 21, 28] proposed RTP approaches that aim at preventing a passive network adversary from reliably distinguishing genuine user activities from "fake" traffic patterns. As shown in Table 1, the general implementation yields additional overhead of 165.9%. Dyer et al. [21] proposed a buffered fixed-length obfuscator based on random padding to prevent website fingerprinting attacks. Juarez et al. [28] proposed an adaptive padding approach that can provide a sufficient level of security against website fingerprinting. The proposed approach matched the gaps between traffic packets with a distribution of generic network traffic. When a large gap is identified, this approach will inject padding traffic in that gap to prevent long gaps from being a distinguishing feature for attackers. Similarly, Apthorpe et al. [9] presented a stochastic traffic padding algorithm to flatten real traffic patterns and randomly inject fake traffic patterns that look like the real IoT traffic patterns. Rather than using pre-defined IoT device traffic pattern distribution, Apthorpe et al. [9] integrated their approach with HMM, which can better model user in-home behavior using IoT traffic trace. However, HMM-based user behavior modeling cannot accurately model user activities that are presented in the interleaved operations of multiple IoT devices simultaneously.

Observation. Our results in Table 1 show that the RTP approach—Tamaraw—yields the lowest PCC, SRCC, and ϵ-security at 0.29%, 0.47%, and 3.4%, respectively. Unsurprisingly, the PTI approach reports the highest PCC, SRCC, and ϵ-security at 0.75%, 0.83%, and 87.15%. This is mainly due to the fact that the PTI approach only injects and adjusts the shape of "fake" traffic patterns and does not reshape or modify any real IoT traffic patterns already presented in IoT traffic traces. The HTR approach reports coarser correlation than the PTI approach. This is because in addition to injecting "noise" into IoT traffic traces, the HTR approach also makes its best efforts to partially flatten both genuine and "fake" traffic patterns of IoT devices. The different correlation performance between the HTR approach and the RTP approach is due to the fact that the RTP approach generally has a higher flattening threshold to pad IoT traffic patterns, and also considers the bidirectional traffic padding for IoT devices (e.g., Amazon Alexa, Google Home). For the same reason, the RTP approach—Tamaraw—reports the maximum traffic overhead of 199% additional overhead per device per day. However, even the best-performing approach—the RTP approach—still reports significant values of PCC and SRCC. This is mainly due to fact that this approach may not consider practical limitations in real smart homes, such as the network bandwidth and maximum traffic injection rate, and thus the "spikes" of genuine traffic patterns can still be observed by adversaries.

2.3 Summary

Prior research proposes significant prevention techniques to thwart privacy attacks on IoT traffic rate traces. Unfortunately, these approaches did not significantly consider at least one of the following facts: (1) the artificial traffic "spikes" that are injected to hide user privacy should not conflict with the real user behavior; (2) many IoT devices have bidirectional network traffic flows that need to be reshaped currently (not necessarily to be perfectly flattened); and (3) flattening and injection operations have practical limitations, such as network bandwidth, maximum package injection rate, and user daily routine permitting. And these may still allow adversaries to infer user in-home sensitive information by applying time series data analytics attacks. In addition, the naive flattening algorithms broadly employed by many approaches actually resulted in three to four times additional traffic overhead. Thus, new lost-cost and effective techniques are necessary. These valuable insights will guide the development of our proposed technique—PrivacyGuard.

3 Privacy Leakage Identification

As discussed in Section 2, we are concerned with sensitive user private information that can be learned by adversaries from externally observed traffic rate traces of IoT devices in smart homes. To explore the severity and extent of this privacy threat, we design a wide range of ML/DL-based, and ensemble method based user activities attack models to better understand and identify the most common user activities that can be learned by these adversaries. Unlike existing work that

Table 2. Best-Performing Attack Models to Detect 13 Different User Activities Using Two Datasets

User Activity	Model	MCC	Cohen's Kappa
Talk to Alexa	Gradient Boosting	0.977	0.955
Control Lights	Decision Tree	0.997	1.000
Print Files	Logistic Regression	0.931	0.933
Baby Present	Random Forest	0.953	0.954
Use Smartphone	SVMs (linear)	0.917	0.942
Use Laptop	Decision Tree	0.997	0.997
Walk in Home	Ada Boosting	1.000	1.000
Check Body Weight	CNNs	0.909	0.999
Check Weather Condition	Random Forest	0.973	0.999
Play Music	LSTM	0.957	0.958
Control Plugs	Passive Aggressive Classifier	0.929	0.927
Make Coffee	SVMs (Linear)	0.969	0.971
Other Activities	LSTM	0.917	0.927

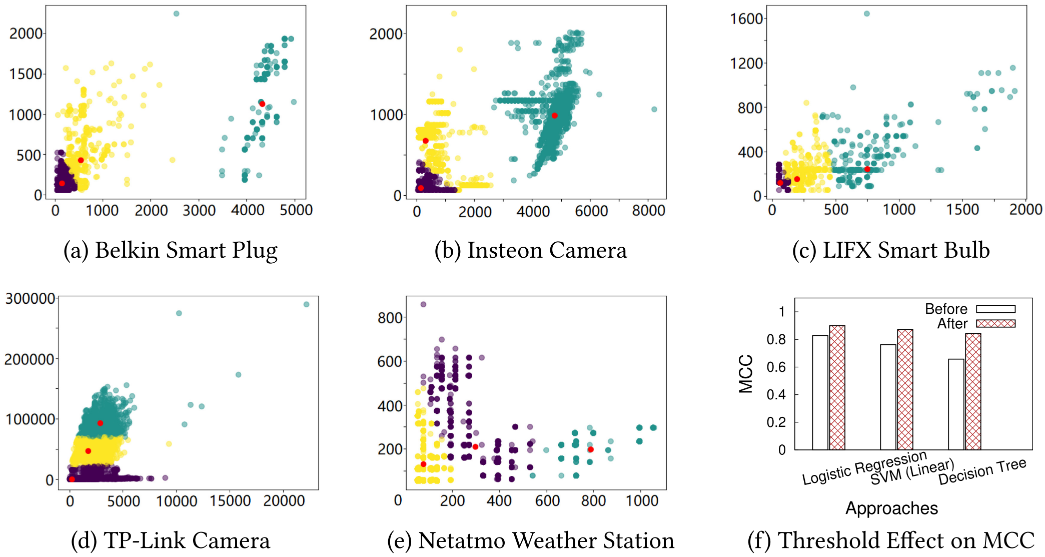
mainly focuses on binary occupancy status detection, we investigate multiple-class (e.g., 13-class classification for Table 2) user activities when a home is occupied. In doing so, we identify the privacy leakage in the traces of the IoT network traffic rate. In addition, we use all of these attack models developed in this section to evaluate our new approach—PrivacyGuard—in Section 6. Note that for each user activity class, we report the attacking model that yields the highest accuracy for each user activity in Table 2. Additionally, we include smarter attack models (e.g., Gradient Boosting, Ada Boosting, and Passive Aggressive Classifier). These complex models are built on top of ensemble methods which could combine the predictions of several base classifiers built with a given learning algorithm to improve generalizability and robustness of adversarial attacks over a single classifier. By doing so, we explore the fuller potential of realistic attackers.

To benchmark the performance of attack models shown in Table 2, we use the **Matthews Correlation Coefficient (MCC)** [34], a standard measure of a binary classifier's performance, where values are in the range −1.0 to 1.0, with 1.0 being perfect user activities detection, 0.0 being random user activities prediction, and −1.0 indicating that user activities detection is always wrong. Cohen's kappa [15] is a measure of the agreement between two classifiers who each classify N items into C mutually exclusive categories. Cohen's kappa is widely used to evaluate multi-class classifiers, where 1.0 indicates a complete agreement, and κ= 0 indicates no agreement among the multi-class classifiers. We will discuss more details about MCC and Cohen's kappa in Section 6.

3.1 Feature Extraction

IoT device traffic events that have user activity information embedded are mainly reflected and captured in the fluctuating spikes or motifs exposed in their traffic rate traces. Next, we will describe our approaches to automatically learn the features that capture user in-home activity information.

3.1.1 Optimal Threshold for Motif Extraction. The first challenge to identify traffic features for attack models is determining the thresholds that we can leverage to filter out background traffic and maximally extract the traffic spikes or motifs. In particular, different IoT devices may require different thresholds due to their different traffic consumption patterns and traffic volumes. For instance, for some on/off IoT devices (e.g., switch, motion/door sensor), which have relatively low traffic demand, we cannot simply use an universal threshold, which may "ignore" user activity information exposed in their traffic motifs.



(a) Belkin Smart Plug (b) Insteon Camera (c) LIFX Smart Bulb

(d) TP-Link Camera (e) Netatmo Weather Station (f) Threshold Effect on MCC

Fig. 5. The illustration of applying K-means on five IoT devices to identify background/low, medium, and high traffic loads.

To address this issue, we design a K-Means clustering-based optimal threshold learning approach. This method aims to identify dynamic thresholds that effectively separate background traffic from all traffic, allowing for the extraction of higher traffic motifs from the background traffic. Our insight is that IoT devices typically have three identifiable traffic consumption patterns, including lower or background, medium, and high traffic mode. As shown in Figure 5(a) through (e), our analytics using the the K-Means clustering algorithm where $K = 3$ has shown that IoT devices typically have three identifiable traffic consumption patterns, including low or background (in purple), medium (in yellow), and high (in cyan) traffic mode. In particular, the volume of background traffic from the same IoT devices, when users are not actively using them, should ideally remain consistent or similar across different smart homes. Thus, we apply the K-Means clustering algorithm where $K = 1$ on the low traffic of all IoT devices to infer the threshold $T_{background}$ that we can leverage to filter out the background traffic which does not capture user in-home activities, and also the medium/high threshold T_{active} that enables our approach to more efficiently extract the user activity embedded traffic motifs. To benchmark the performance of optimal threshold searching, we examine three different attacks using the user activity of talking to Alexa utilizing the attack models shown in Table 2. We report the top-3 attack models, including Linear Regression, SVM, and Decision Tree, which yielded the best attacking performance in Figure 5(f). We find that employing optimal thresholds to attack user privacy yields better MCC results compared to using a static and universal threshold. That being said, optimal thresholds could enable attackers to more efficiently infer user private information embedded in IoT traffic rate traces. The algorithm for optimal threshold searching is established in Algorithm 1.

3.1.2 Principle Feature Identification from Motifs. The second challenge is to learn the principle features from these extracted traffic motifs. We leverage the **Principal Component Analysis (PCA)** algorithm to identify the principle features that we can use to build ML-based and DL-based smart attack models. We first build a large IoT traffic rate dataset that has network traffic rate traces of 31 IoT devices and empirically examine 10 statistical features based on the time series motifs of

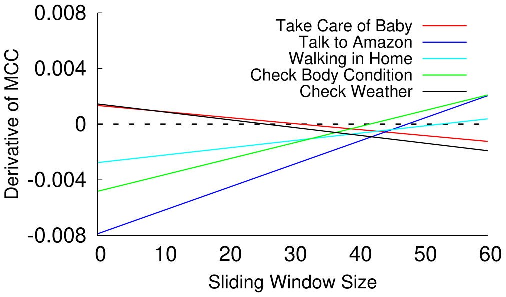


Fig. 6. The relationship between sliding window and user activity inference accuracy.

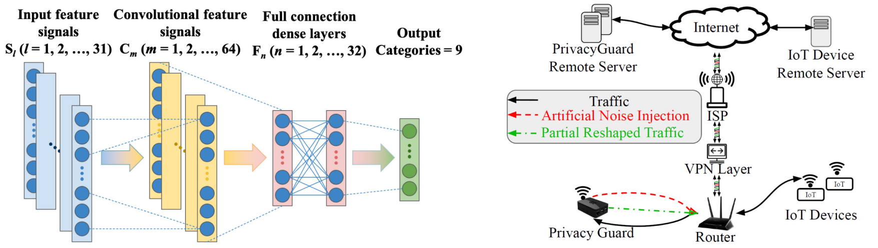
ALGORITHM 1: Optimal Threshold Learning

```
Input: Traffic Volume V, Device List L
Output: Threshold T
Data: Traffic Volume V, Device List L, Threshold T
$V(t, t+\delta) = \{V_t, V_{t+1}, \ldots, V_{t+\delta}\}$ where $\sum_{t=1}^{V_t} := V$
/* For each IoT device, infer the low, medium, and high thresholds. */
for $\forall\ i \in L$ do
    $\{T_{Low_i}, T_{Medium_i}, T_{High_i}\}$ = $K$-Means ($V_i(t, t+\delta)$, K=3)
Set $T_{background}$ = $K$-Means ($T_{Low}, K = 1$)
Set $T_{active}$ = $[\min\{T_{Medium}, T_{High}\}, \max\{T_{Medium}, T_{High}\}]$
for $\forall\ V(t, t+\delta) \in V$ do
    if $V_t \gtrsim T_{background}$ then
        Continue
    else
        /* Reshaping traffic load using selected threshold */
        $V_t = T_{background}$
```

each IoT device, including duration, mean, maximum and minimum values, standard deviations, range, skewness, variation coefficient, kurtosis, and AUC (area under the curve), among others. We leverage PCA to analyze the principle features from IoT network traffic rate traces.

We then further process these traffic spikes using a sliding window to learn the sequential event characteristics that may appear in bidirectional traffic IoT devices. For instance, Amazon Echo typically presents a short burst of outgoing traffic and then incoming traffic flows in its traffic rate trace. In addition, user activity events usually have different duration which may affect the prediction performance of attack models. Given a specific traffic rate trace, we extract the whole traffic into multiple independent spikes that can be potentially employed to identify different user activities. We then learn the preceding statistical metrics using a sliding window n. To ensure the effectiveness of all attack models on different user activities, we need to find the optimal sliding window size n that can accommodate all IoT devices. Figure 6 presents the derivative of 2-degree polynomial fitting of the attack model performance. As shown in that figure, the sliding window size has a significant effect on the accuracy to identify different user activities. We find the optimal sliding window size $n = 40$ that we can guarantee our inference attack models could observe and learn the principle features exposed in traffic rate spikes to indicate user in-home activities.

Note that the granularity of traffic rate traces also significantly impacts the performance of the selected features. For instance, for lower/coarser granularity traffic traces, some features such


Input feature signals
S_l (l = 1, 2, ..., 31)
Convolutional feature signals
C_m (m = 1, 2, ..., 64)
Full connection dense layers
F_n (n = 1, 2, ..., 32)
Output Categories = 9
PrivacyGuard Remote Server
Internet
IoT Device Remote Server
Traffic
Artificial Noise Injection
Partial Reshaped Traffic
ISP
VPN Layer
IoT
IoT
IoT Devices
Privacy Guard
Router

(a) The overview of our CNN architecture (b) System model of PrivacyGuard

Fig. 7. The overview of PrivacyGuard's CNN structure (a) and system model (b).

as duration, standard deviations, AUC) might be less distinguishable and hidden, and thus the ϵ-security of the external adversaries' attack models will significantly decrease. For example, considering user activities that have a very short duration, normally at the second-level (e.g., operating switches), lower/coarser granularity traffic traces may hide the traffic features and significantly lower down the performance of attack models. Similarly, for long-lasting activities that are minute-level or longer, (e.g., checking body condition), traffic traces at lower/coarser granularity can preserve traffic rate signature and their features better. A fuller evaluation of granularity effect is discussed in Section 6.4.4.

3.2 ML-Based Inference Attacks

We then focus on selecting the optimal ML model that has the best accuracy to detect user activity. We investigate the most widely used ML classifiers in prior IoT traffic research work, including Logistic Regression, SVMs, and Random Forest. In particular, we also benchmarked different kernels for SVMs, including linear, linear passive-aggressive, linear ridge, polynomial with 1∼10 degrees, and radial basis function. Table 2 shows the results for attacking 13 different user activities. Note that for each user activity shown in Table 2, we run all of the ML-based attack models and report the one that has the best attacking accuracy in MCC and Cohen's kappa.

3.3 DL-Based Inference Attacks

In addition to ML-based attack models, we design a CNN-based DL approach to detect user in-home activities from IoT traffic rate traces. In the following, we describe the design of our CNN architecture, which is inspired by the most notable prior CNN research—VGGNet [52]. As shown in Figure 7(a), our CNN architecture is composed of input, convolutional layers (ReLU), max pooling, fully connected layers (with and without ReLU), and output. In addition, two fully connected layers with ReLU and another fully connected layer (without ReLU) are added to process the output.

3.4 Comparison and Summary

Interestingly, as shown in Table 2, it is surprisingly easy to infer and learn user in-home activities using their network traffic rate traces in a smart home. On average, our ML-based and DL-based attacking approaches yield the average MCC as 0.956 and the average Cohen's kappa as 0.966. These results show that our implemented ML-based, DL-based, and ensemble method based attacks are effective at detecting a user's private sensitive information (e.g., user activities) in a smart home. Specially, the AdaBoost-based attack model achieves the best inference attack results. Thus, IoT traffic rate traces expose a serious threat to user in-home privacy. Therefore, new privacy preserving techniques are necessary. We employ all of the preceding attack models to evaluate our new approach—PrivacyGuard—in Section 6.

4 PrivacyGuard Design

In this section, we explain how we design PrivacyGuard, a new defense system that enables users to enjoy the benefits offered by IoT devices while also controlling the privacy of their traffic data with reasonable traffic overhead.

4.1 Approximate Differential Privacy

Differential privacy was first presented in the work of Dwork [19] to measure the individuals' privacy loss on database queries. Recently, differential privacy algorithms have become a trend in the privacy research community of IoT devices and smart homes [20, 26, 42, 58, 58, 59], and have been broadly adopted by Apple [4], Google [5], and Amazon [1]. Given a smart home system, we can say a defense approach that can ensure if an arbitrary single substitution in the IoT traffic rate traces is small enough if the statistical query learning results cannot be used to infer accurate user in-home sensitive information in a smart home, and thus preserves the user privacy that may be exposed in the smart home network traffic rate traces. Typically, (ϵ)-differential privacy is used as a formulated metric to describe the privacy guarantee. Lower values of ϵ indicate a stronger guarantee in a defense system. The definition for (ϵ)-differential privacy is this: *An algorithm A is (ϵ)-differential private if for all traffic trace substitutions T_1 and T_2 where T_1 and T_2 differ by at most one traffic rate signature, and for all subsets of possible answers $S \subseteq Range(A)$:*

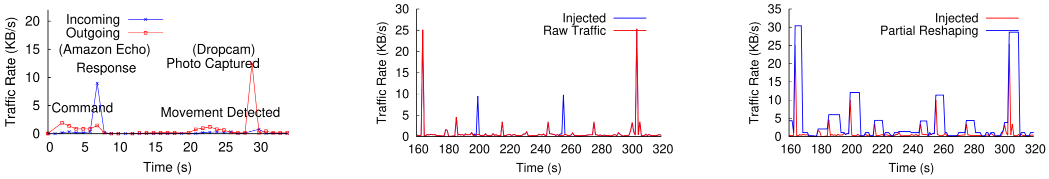
$$P[A(T_1) \in S \leq P[A(T_2) \in S] \cdot exp(\epsilon). \tag{1}$$

If the inequality is satisfied, then the defense approach A's output is considered to be ϵ indistinguishable and it will be hard for an external attacker to perform a traffic analytics attack on the smart home IoT traffic. Dwork [20] proposed an approximate version of differential privacy, which can be described as follows:

$$P[A(T_1) \in S \leq P[A(T_2) \in S] \cdot exp(\epsilon) + \delta. \tag{2}$$

Approximate differential privacy is a relaxed version of standard differential privacy. The parameter δ enables the algorithm A to not be the ϵ differential privacy for some portions of IoT network traffic. Our system is motivated by this approximate differential privacy. Given a target smart home, T_1 and T_2 are two substitutions of IoT network traffic rate traces in a smart home, and at most one traffic rate signature/spike is different. An external attacker is trying to identify principle network traffic features which are processed by applying the algorithm A and thus to predict the associated user in-home activity. For approximate differential privacy with parameters (ϵ, δ), it is hard for an external attacker to perform traffic analysis attack to infer users' in-home activities. Our evaluations have shown that PrivacyGuard could achieve the approximate differential privacy.

Please note that although differential privacy is stronger and more desirable than approximate differential privacy, achieving the latter is more system practical and can still could be effective for us to build a low-cost computer system to help users safeguard their IoT device traffic. Our focus is to design a new low-cost and user-centric defense computer system that enables people to effectively regain the privacy leakage control of their IoT devices. PrivacyGuard preserves user approximate differential privacy by combining intelligent traffic rate signature learning, artificial traffic rate signature injections, and partial traffic reshaping to approximate the algorithm A. Our system design and implementation incorporates the idea of approximate differential privacy due to its natural insight aligning with our problem requirements. We also understand that there are some defense limitation studies on differential privacy [14, 27]. However, the full theoretical proof of differential privacy and its limitation study is outside the scope of this work. In addition, our system design is orthogonal to new versions of approximate differential privacy. Users have the



(a) Traffic Signature Learning (b) Artificial Traffic Signature Injection (c) Partial Traffic Reshaping

Fig. 8. PrivacyGuard's traffic signature learning (a), artificial traffic signature injection (b), and partial traffic reshaping (c). Keyang added four new complex models.

flexibility to "plug in" new approximate differential privacy ideas to our open source platform by tuning our thresholds and other parameters, potentially benchmarking and enhancing their system performance.

4.2 System Design

Figure 7(b) shows the system structure of our PrivacyGuard, which assumes that either a software **Virtual Private Network (VPN)** or hardware VPN router is deployed in a smart home. PrivacyGuard is then connected to the Wi-Fi access point, such as home router or home gateway. Note that PrivacyGuard can be deployed either on an IoT hub (shown in Figure 7(b)), home router, or Wi-Fi gateway. A VPN wraps all smart home traffic from IoT devices in an additional transport layer. By doing so, the VPN can aggregate all traffic into a single traffic flow with the source and destination addresses of the VPN endpoints. Our proposed new approach—PrivacyGuard—allows user "tunable" control over what can be learned using data analytics techniques over traffic rate traces from a smart home. PrivacyGuard leverages the VPN layer as the first defense to prevent user in-home activity inference, although even if the VPN has been optimally configured, the external adversaries may still be able to infer user activities due to user sparse activity and dominating IoT devices [9]. Then, PrivacyGuard takes additional actions to further obscure user in-home privacy. In essence, PrivacyGuard first learns IoT device traffic signatures from their historical traffic data. Then, PrivacyGuard employs a DL-based user in-home activity modeling to inject artificial traffic signatures into traffic rate traces such that the genuine user traffic signatures are obscure in the modified traffic rate traces. Next, PrivacyGuard partially reshapes IoT device' traffic rate traces by considering practical limitations. In addition, PrivacyGuard employs multiple optimization techniques to further obscure the privacy information that are exposed in the externally observed traffic rate traces with lower traffic overhead. The overhead (e.g., additional traffic flows) is reduced due to our careful design to learn and reshape traffic signatures in our system. Figure 8 shows the three major operation flows of PrivacyGuard. Note that end users are not required to (frequently) retrain or manage the components in our system pipeline. Instead, users can use PrivacyGuard in default settings (with pre-trained models).

Note also that users are not obligated to provide input during the setup process or retrain their models within our system. For those with heightened privacy protection needs, they can customize preferences to tune our traffic reshaping thresholds. Our system offers users the flexibility to choose their desired balance between data utility and user privacy.

4.3 Intelligent Traffic Rate Signature Learning

PrivacyGuard first learns IoT device traffic rate signatures that are used in its later traffic reshaping algorithms. The goal of this traffic rate signature learning is to ensure that it is reliably difficult for the external adversaries to distinguish the genuine IoT traffic rate signatures from the "artificial"

ALGORITHM 2: Traffic Signature Learning

```
Input: Traffic Volume $V$
Output: Traffic Signature $S$
Data: Traffic Volume $V$, SQLite Signature Database $DB$
/* Segment aggregated traffic volume into device levels */
Disaggregate traffic volume $V$ into device $i$'s volume $V_i$
for $\forall V_i \in V$ do
    if $Duplicated\_Signature\ (V_i)\ in\ DB$ then
        /* Similar traffic signature already exists */
        Continue
    else
        /* New traffic signature found */
        Insert $V_i$ into $DB$
        Update index_keys of $DB$
/* Learn device appearance pattern */
for $\forall T_i \in T$ do
    for $Day_i \in [0, 6]$ do
        for $Hour_i \in [0, 23]$ do
            $TC_i = TC_i + 1$          // Update traffic frequency
            $TV_i + = TV_i$            // Update traffic rate
```

injected or replayed traffic rate signatures. Different IoT devices typically have different traffic signatures. For a specific IoT device, PrivacyGuard can learn its traffic rate signatures over time both offline and online. Figure 8(a) shows the traffic rate signatures (in kilobytes per second) of Dropcam and Amazon Alexa. We store all traffic signatures for IoT devices in an SQLite database. PrivacyGuard also takes additional steps to ensure that it is reliably difficult for external adversaries to distinguish artificial traffic demand from real traffic demand in the SQLite database. For instance, the time and duration for each traffic signature, and also other attributes, such as short, long, high, low, and medium, may compute the fraction of traffic signatures in each category. Then, we use this fraction to weight each category's future traffic signature selection such that the "artificial" traffic demand matches the breakdown of real traffic demand. PrivacyGuard uses the PCC [40], which is a measure of the linear correlation between the current traffic rate signature and old traffic rate signatures to eliminate the duplicated traffic rate signature update. PrivacyGuard examines the incoming traffic rate signatures in the same manner, despite whether they are "old" or "new" traffic patterns. The major difference is that once a new signature is detected, we keep a copy in our database for signature learning and future injection usage. Similarly, PrivacyGuard can also detect and replay the new traffic rate signatures generated by the "old" devices. The algorithm for traffic rate signature learning is established in Algorithm 2.

In addition, we observed that some IoT devices (e.g., body condition measurement devices and smoke sensors) have much less frequent daily usages than other intensive user interaction IoT devices. Thus, to ensure the accuracy and quality of traffic rate signature learning for these IoT devices, we leverage DCGANs [43] to build a new traffic rate signature generator to enrich the training traffic data samples for those small traffic devices. Our DCGAN architecture is composed of convolutional layers without max pooling or fully connected layers. We leverage convolutional stride and transposed convolution for downsampling and upsampling, respectively. The generator network uses a 100*1 noise vector. Our first layer is to project and reshape inputs, then following

ALGORITHM 3: Artificial Traffic Signature Injection

```
Input: Traffic Volume $V$
Output: Traffic Signature $S$
Data: Traffic Volume $V$, SQLite Signature Database $DB$
/* Inject artificial traffic signatures                                */
for $Day_i$  $[0, 6]$ do
    for $Hour_i \in [0, 23]$ do
        if $TC_i \gneq 0$ then
            $TC_i - = TC_i$                        // Update traffic frequency limit
            $TV_i - = TV_i$                             // Update traffic rate limit
            /* Mimic user activities using LSTM                        */
            Select traffic signature $\overline{V_i}$ from $DB$ based on our learned user activity $H$
            /* Further obscure privacy in the load                     */
            Update traffic volume $V_i = V_i + \overline{V_i}$
```

this layer, we have five convolutional layers. For the generator model, we use the ReLU activation function for all layers except the final one, where we employ the Tanh activation function. Our generator and discriminator have almost the same architectures, but reflected. For the discriminator model, we use the Leaky ReLU activation function for all layers except the last layer where we use the Sigmoid activation function. By doing this, we are able to build a rich set of traffic rate signatures for these IoT devices. Note that learning a traffic rate signature does not necessarily mean that PrivacyGuard will inject it. The injecting decisions are made by our real user behavior modeling based traffic signature injection process, which will be explained in the next section.

4.4 Artificial Traffic Signature Injection

PrivacyGuard does not simply inject or replay traffic signatures randomly, since an external adversary may be able to identify those random patterns in smart home traffic rate traces. This may still allow external adversaries to distinguish the injected "fake" traffic demand patterns from the real traffic demands due to their inconsistency in user in-home behaviors in a specific smart home.

Prior approaches have explored the benefits of integrating real user behavior with their privacy preserving approaches using Bernoulli distribution, Poisson distribution, or **Linear Chain Conditional Random Field (LCCRF)** into their traffic "noise" injections into IoT traffic traces. PrivacyGuard selects signatures from the database to inject at an injection rate equal to the rate at which the home generates traffic rate traces when occupied. In addition, PrivacyGuard injects realistic traffic signatures that we learn from real IoT device traces in Section 4.3. More importantly, PrivacyGuard considers real user behaviors in a smart home when injecting these realistic traffic rate signatures for each IoT device. In doing so, PrivacyGuard can ensure that the injected traffic patterns still fit the traffic distributions that represent the regular user in-home behaviors such that the external adversaries cannot distinguish injected traffic patterns from genuine traffic patterns. The algorithm for traffic rate signature injection is established in Algorithm 3. Next, we explain how PrivacyGuard models user in-home activities.

LSTM-Based User In-Home Activities Modeling. To address this problem, we present a recurrent neural network based approach to model real user in-home behaviors. Specifically, we design an LSTM-based approach to model user in-home behavior using IoT traffic rate traces. Note that similar to HMM, the LSTM-based approach also assumes that user activities behind these IoT device events are hidden and thus can be learned through the LSTM architecture. Compared with the

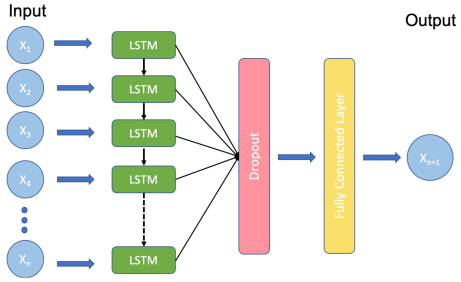
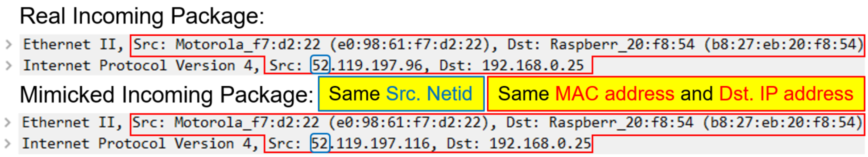


Fig. 9. Overview of LSTM-based user in-home daily routine modeling.

HMM-based approach, our LSTM-based model can learn user in-home activity using both single IoT device events and the concurrent events of multi-IoT devices. The input shape (a.k.a. window size) of our LSTM model is the status vector size of all IoT devices, and the IoT devices' reported sensor data is associated with nine different user in-home activities. The output of the LSTM model is the future user activities. As shown in Figure 9, the first visible layer is the LSTM layer with 10 x 10 memory blocks. To reduce overfitting and improve model performance, we apply 20% dropout to the recurrent input signals on the LSTM units. After that, two fully connected layers with ReLU and another fully connected layer (without Softmax) are added to process the output. Since PrivacyGuard is performing multi-class user activity classification, we use Categorical Cross-Entropy Loss (a.k.a. Softmax Loss) as model loss function. In addition, instead of using the classical stochastic gradient descent approach to update the parameter weights, we employ the Adam algorithm as the optimizer for our LSTM model that can better handle high-dimensional parameters and mitigate sparse gradient problems. To train our LSTM model, we split IoT device traces with a 70%/30% split of training data to test data. PrivacyGuard leverages the LSTM-based user activity model to select what IoT traffic rate signatures to inject and when to inject them. Note that user daily routines, user populations, and user patterns may be different in different homes. In addition, in a new home, user activity, home configuration, and IoT devices may vary. Users can deploy our PrivacyGuard to automatically retrain the preceding LSTM model to learn these user in-home patterns which we benchmarked in Table 2.

Bidirectional Traffic Signature Injection. The way that PrivacyGuard leverages to mimic unidirectional communication IoT devices is trivial. However, a significant amount of IoT devices are user interaction intensive, such as voice assists and IoT smart plugins, and they have bidirectional traffic flows. To mimic these IoT devices, as shown in Figure 7, PrivacyGuard may be deployed both locally and on the remote servers using a Master/Slave model. The local PrivacyGuard works regularly as the master which is very similar to other single directional traffic IoT devices, whereas the remote PrivacyGuard server acts as the remote IoT device servers that are responding to local IoT device traffic demands. In addition, PrivacyGuard works in a mixed architecture of Master-Slave and Publish-Subscribe. The remote servers have the same design as the local PrivacyGuard. The mapping relationship between local in-home PrivacyGuard (a.k.a. publishers) and remote PrivacyGuard servers (a.k.a. subscribers) is N:M. In other words, multiple PrivacyGuards can share a remote PrivacyGuard server, and a single smart home PrivacyGuard can be paired with at least one remote server. The remote server is pretending to be the "valid" IoT remote server to respond to artificial IoT device bidirectional traffic demands. To mimic the incoming/inbound traffic, we build the PrivacyGuard remote server on top of the traffic and package editor/generator—Ostinato [2]—that supports most common standard protocols including Ethernet/802.3/LLC, VLAN, ARP, IPv4, IPv6, TCP, UDP, HTTP, SIP, RTSP, and NNTP. In particular, PrivacyGuard leverages the Ostinato



Real Incoming Package:

```
> Ethernet II, Src: Motorola_f7:d2:22 (e0:98:61:f7:d2:22), Dst: Raspberr_20:f8:54 (b8:27:eb:20:f8:54)
> Internet Protocol Version 4, Src: 52.119.197.96, Dst: 192.168.0.25
```

Mimicked Incoming Package:

```
> Ethernet II, Src: Motorola_f7:d2:22 (e0:98:61:f7:d2:22), Dst: Raspberr_20:f8:54 (b8:27:eb:20:f8:54)
> Internet Protocol Version 4, Src: 52.119.197.116, Dst: 192.168.0.25
```

Fig. 10. The illustration of PrivacyGuard remote server modified packages for Amazon Echo.

Python API [2] to vary packet fields across packets at runtime—for example, changing the source IP/MAC addresses in the packages of the PrivacyGuard remote server to those of the actual IoT remote server (shown in Figure 10). In doing so, PrivacyGuard is able to generate incoming traffic from the source "valid" IoT remote server. In addition, using this design, a single point of remote server failure will not prevent PrivacyGuard from injecting artificial incoming traffic that is critical to hide user privacy in traffic traces. Note that the possible extra traffic from/to cloud servers, such as copy-cat traffic pattern injections, may serve as "free" noise injections and actually can help PrivacyGuard better hide user sensitive information in the traffic rate traces.

4.5 User-Centric Partial Traffic Reshaping

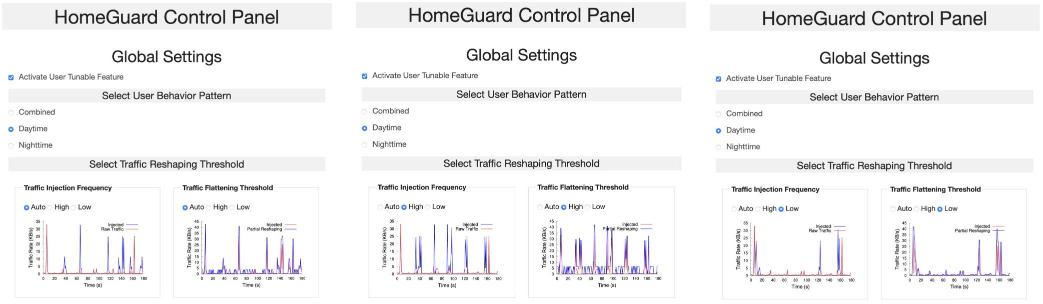
After applying the LSTM-based artificial traffic signature injection, the modified traffic traces may still expose changes in traffic rate spikes. To hide these remaining spike changes, we design a new user tunable partial traffic reshaping approach. Unlike prior approaches [9, 12, 16, 37, 54–57], simply assuming that their reshaping techniques always have enough or unlimited traffic bandwidth to completely flatten the spikes in the externally observed traffic traces, PrivacyGuard employs a reshaping threshold $T_{reshape} = \max\{T_{current}(t), U(t), T_{average}(t)\}$ that only partially reshapes the traffic demand to a target less than the peak traffic demand. $T_{current}(t)$, $U(t)$, and $T_{average}(t)$ denote the current traffic rate demand, the user preferred set point, and the average traffic rate demand, respectively. To maintain $T_{reshape}$ at each t with current traffic demand $T_{current}(t)$, PrivacyGuard consumes $T_{reshape} - T_{current}(t)$ whenever $T_{current}(t) < T_{reshape}$. Since $T_{reshape}$ traffic demand is typically much lower than peak traffic demand, a low reshaping threshold is able to hide the most of the changes in traffic rate trace data without using much network bandwidth. The algorithm for user tunable partial traffic reshaping is established in Algorithm 4.

Figure 11 illustrates the results of PrivacyGuard when smart home users set their privacy guarantee to "Auto," "High," and "Low" preferences. Note that under the "Auto" mode, PrivacyGuard can automatically learn an optimal/default tradeoff point such that users can use the "least" traffic overhead to protect their smart home from the "most" privacy leakage. In addition, PrivacyGuard supports smart home users, such as those who require more privacy protection or are on an unlimited Internet data plan, to "tune" this learning process such that they can use more traffic to hide their privacy information exposed in their traffic rate traces. Through this module, smart home users could regain the control of at what degree the users would like to manage their privacy leakage from IoT traffic rates.

4.6 Online Optimizations

In addition, PrivacyGuard introduces some optimization techniques to further obscure the potential privacy leakage in the externally observed traffic rate traces, including intelligent traffic signature adjustment and random noise injection, and reshaping rate adjustment. We describe the detail of each optimization as follows.

Intelligent Traffic Signature Injection Adjustment. PrivacyGuard adjusts the replayed signature by raising or lowering each point by a small random amount (e.g., 0% to 5% of traffic demand).



HomeGuard Control Panel

Global Settings

☑ Activate User Tunable Feature

Select User Behavior Pattern

○ Combined
● Daytime
○ Nighttime

Select Traffic Reshaping Threshold

(a) Traffic Injection Frequency: ● Auto ○ High ○ Low — Traffic Flattening Threshold: ● Auto ○ High ○ Low

(b) Traffic Injection Frequency: ○ Auto ● High ○ Low — Traffic Flattening Threshold: ○ Auto ● High ○ Low

(c) Traffic Injection Frequency: ○ Auto ○ High ● Low — Traffic Flattening Threshold: ○ Auto ○ High ● Low

(a) Automatic (b) High reshaping rate (c) Low reshaping rate

Fig. 11. PrivacyGuard's user tunable traffic reshaping GUIs, including automatic self-learning mode (a), high mode (b), and low mode (c).

ALGORITHM 4: Partial Traffic Reshaping

```
Input: Traffic Volume $V$, User Preference $U$
Output: Modified Traffic Volume $V$
Data: Traffic Volume $V$
/* Segment traffic volume trace into isolated traffic traces */
Separate traffic volume $V$ into time $t$'s volume
$V(t, t+\delta) = \{V_t, V_{t+1}, \dots, V_{t+\delta}\}$ where $\sum_{t=1}^{V_t} := V$
for $\forall\ V(t, t+\delta) \in V$ do
    Set Reshaping_threshold = $\max\{TV_t, U, Current\_load_t\}$
    if $V_t \geq Reshaping\_threshold$ then
        Continue
    else
        /* Reshaping traffic load using selected threshold */
        $V_t = Reshaping\_threshold$
        Extend the reshaping for random $\epsilon$ seconds
        $V(t, t+\epsilon) = Reshaping\_threshold$
```

In addition, for each traffic rate demand reshaping, PrivacyGuard extends its duration by a small random amount (e.g., 0% to 5% of the regular duration) such that the starting and ending points in the traffic rate signature of the sleep sensor like IoT devices are hidden. PrivacyGuard only injects traffic rate signatures when the home user behaviors permit. For instance, at nighttime, when most smart home users are sleeping, PrivacyGuard needs to ensure that traffic traces have significant less interactive IoT device traffic demands.

Traffic Injection Rate Adjustment. PrivacyGuard also dynamically adjusts its reshaping threshold and rate of artificial traffic rate signature injection over time to match the expected rate each period. Our insight is that there is no need to make lower-traffic nighttime periods look like high-demand traffic daytime periods. Instead, PrivacyGuard only ensures that these time periods look the same with respect to each other, regardless of whether a home is occupied or not. In addition, PrivacyGuard indexes its traffic rate signatures database based on each IoT device's traffic rate signature's real time of use. At any time, PrivacyGuard is trying to select from the past traffic rate signatures that occurred near that time when the LSTM-based user behaviors model allows.

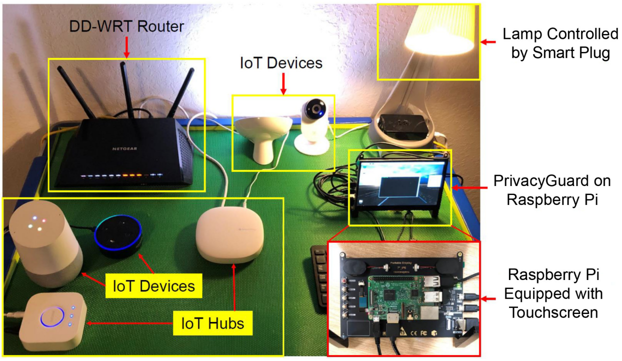

DD-WRT Router
IoT Devices
Lamp Controlled by Smart Plug
PrivacyGuard on Raspberry Pi
Raspberry Pi Equipped with Touchscreen
IoT Devices
IoT Hubs

Fig. 12. The overview of our PrivacyGuard prototype.

ML and DL Model Improvements. Finally, PrivacyGuard is orthogonal to the specific ML and DL technique. For instance, we use SVM, Random Forest, Logistic Regression, and other models to build smart external attacks to evaluate existing defense performance. Our goal is to demonstrate that it is quite straightforward to infer user in-home activities. As another example, our system design incorporates LSTM due to its natural architecture aligning with our problem requirements. Users have the flexibility to "plug in" new models to our open source platform, potentially benchmarking and enhancing system performance.

5 Implementation

We implement PrivacyGuard, both simulator and prototype, in `python` using widely available open source frameworks, including Pandas, Scikit-learn, and PyCUDA. The simulator takes a home's network traffic race traces as input and applies privacy preserving techniques outlined in the previous section. We also deploy a prototype PrivacyGuard in a "mock"smart home to demonstrate the ability to modulate a home's network traffic rate demands in real time to mask user activities using PrivacyGuard's approach online. As shown in Figure 12, we employ a Raspberry Pi 4 Model B based hardware (Broadcom BCM2711, Arm Cortex-A72 Architecture) setup, which enables PrivacyGuard to reshape, inject, and adjust traffic rate demands in real time. The prototype uses IoT network traffic rate data at the home's Wi-Fi access points to query the real-time traffic rate readings for the entire home every minute using `cronjobs`. We implement PrivacyGuard's algorithms and its optimizations. We deploy our PrivacyGuard remote server on the Amazon EC2 t1.micro instance with a cost of \$0.0035 per hour. We also store the set of artificial traffic rate signatures, indexed by time period, that are available for replay in an *SQLite*3 database. The size of the implementation is less than 1,500 lines of code. We use the Scikit-learn ML library in `Python` to build our ML attack approaches. The library supports multiple techniques including Logistic Regression, SVMs, and Random Forest. In particular, we also implemented different kernels for SVMs, including linear, linear passive aggressive, linear ridge, polynomial with 1∼10 degrees, and radial basis function, and PCA. For CNN-based attack approaches, we implement based on the framework from VGGNet [52]. For user in-home activities, we implement LSTM-based user in-home activities modeling using the Keras model library [30] and TensorFlow framework [6]. Finally, we schedule the batch jobs on our GPU servers to compare the MCC accuracy of eight different approaches using CUDA. The server we use to get all benchmarking and evaluation results for attacking models has the following resources: (1) CPU: 2x Intel Xeon CPU E5-2620 v4 @ 2.10 GHz, (2) GPU: NVIDIA TITAN X (Pascal) (x8), (3) RAM: 128 GB, and (4) OS: Linux CentOS 7. PrivacyGuard can be implemented on IoT hubs and middle boxes (e.g., Wi-Fi access points, gateway routers, and smart IoT hubs).

6 Experimental Evaluation

Next we describe our datasets, experimental setup, metrics used to evaluate our PrivacyGuard approaches, and evaluation results.

6.1 Datasets

Dataset 1: UNSW. We downloaded the publicly available IoT traffic rate traces from UNSW Sydney [48] that include packet level network traffic traces of 22 IoT devices for 20.5 days. These raw traffic traces contain packet headers and payload information. To evaluate our approaches, we pre-process the IoT traffic metadata traces to IoT traffic rate data at different granularities and also label all user in-home activities.

Dataset 2: SmartFIU. We set up our own "mock"smart home using our laboratory space that has four graduate students operating 31 IoT devices daily. We first deploy a NETGEAR AC1750 smart Wi-Fi router that serves as the internal switch and the gateway to the public Internet. We flash the router using DD-WRT [17] (a Linux based alternative open source firmware) and set up ExpressVPN [22] on the router. We then install *tcpdump* on this gateway to capture network traffic data. We use 45 days of 1 minute level IoT traffic data to evaluate our PrivacyGuard.

Dataset 3: Real Smart Home Dataset. We deploy 22 popular IoT devices in two real smart homes. The two homes are private townhouse apartments that have two and four occupants, respectively. We collect the second level IoT traffic traces of 22 IoT devices from the two homes, resample them into IoT network traffic traces of different granularities, and also record log groundtruth user activities for the whole 2 weeks.

Dataset 4: IoT Device Captures (Kaggle #1). We download the IoT Device Captures dataset from Kaggle, which has 30 popular IoT devices. Each IoT device was recorded for 20 segments, and each segment has a traffic duration as of 2 minutes. We label the groundtruth user activities by examining their IoT device events.

Dataset 5: IoT Device Network Logs (Kaggle #2). We also download the IoT Device Network Logs dataset, which captured 1 minute level network traffic traces of 14 popular smart home IoT devices for 5 days using NodeMCU with an ESP8266 Wi-Fi module. We label the groundtruth user activities by examining their IoT device events.

Note that to label user activity in public datasets rather than ours, we develop a script to assist us to search motifs in aggregated traffic spikes. Then, we cluster and process the groundtruth user activities data comprehensively. For our own datasets, we have been logging user activities in our monitoring smart homes. In addition, to learn the effect of traffic rate granularity on the user privacy preserving degree, we pre-process the traffic rate traces of the preceding datasets into different granularity levels (e.g., 1 second, 1 minute, 3 minutes, 5 minutes, and 10 minutes). By default, traffic rate granularity is set at 1 second.

6.2 Experimental Setup

PTI Approach. We first implement a general version of prior work [12, 39]. This approach leverages Bernoulli distribution, Poisson distribution, and LCCRF to randomly inject "fake"traffic demands that are randomly selected from historical traffic patterns.

HTR Approach. We implement a general version of prior work [10, 13]. This approach employs a threshold-based traffic demand flattening, and leverages Bernoulli distribution, Poisson distribution, and LCCRF to randomly inject "fake" traffic demands.

RTP Approach. We implement a general version of prior work [9, 21, 28]. This approach employs traffic demand flattening and leverages HMM-based user behavior modeling to randomly inject “fake”traffic demands that are randomly selected from historical traffic patterns.

PrivacyGuard Approach. PrivacyGuard employs intelligent DCGAN-based IoT device traffic signature learning, LSTM-based artificial traffic signature injection, and partial traffic reshaping to further obfuscate private information that can be externally observed in IoT traffic traces. We also evaluate our PrivacyGuard with the online optimization approaches as we discussed in Section 4 to further obscure user private information in the externally observed traffic rate traces, including intelligent traffic signature adjustment and traffic rate adjustment.

6.3 Evaluating Metrics

Next we describe the metrics that we use to evaluate PrivacyGuard.

Matthews Correlation Coefficient. To quantify the accuracy of different user privacy enhancing approaches, we note that the standard evaluating metrics (e.g, accuracy and F1) would not work well on our highly imbalanced IoT traffic data. Based on the recommendation from prior work [7, 41], we use the MCC [34], a standard measure of a classifier’s performance, where values are in the range from −1.0 to 1.0, with 1.0 being perfect user activity detection, 0.0 being random user activity prediction, and −1.0 indicating that user activity detection is always wrong. The expression for computing MCC is as follows, where TP is the fraction of true positives, FP is the fraction of false positives, TN is the fraction of true negatives, and FN is the fraction of false negatives, such that TP + FP + TN + FN = 1:

$$\frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{3}$$

Cohen’s Kappa. Cohen’s kappa [15] is a measure of the agreement between two classifiers who each classify N items into C mutually exclusive categories. Cohen’s kappa is defined as

$$\kappa = 1 - \frac{1 - p_o}{1 - p_e}, \tag{4}$$

where p_o is the relative observed agreement among classifiers, and p_e is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each classifier randomly seeing each category. If the classifiers are in complete agreement, then κ should be 1. If there is no agreement among the classifiers other than what would be expected by chance, then $\kappa = 0$.

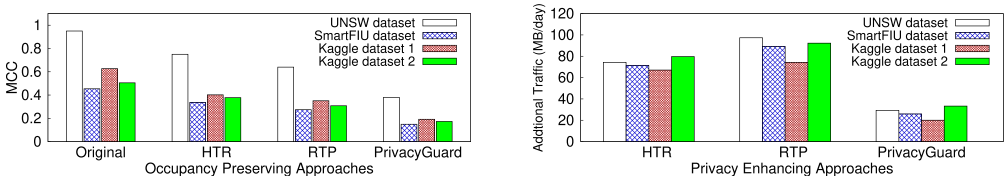
Pearson Correlation Coefficient. The PCC [40] is a measure of the linear correlation between two variables (e.g, original and modified traffic), computed as the covariance between the variables divided by the product of their standard deviation. It has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation.

Spearman’s Rank Correlation Coefficient. The SRCC [49] between two variables is equal to the PCC between the rank values of those two variables (e.g, original traffic and modified traffic). However, unlike the PCC that assesses linear relationships, the SRCC assesses monotonic relationships (whether linear or not). If there are no repeated data values, a perfect SRCC of +1.0 or −1.0 occurs when each of the variables is a perfect monotone function of the other. We use the PCC and the SRCC to quantify the effectiveness of different approaches on masking user private information.

Adversary Confidence. We leverage **Adversary Confidence (AC)** to describe the adversary’s ability to identify which time periods are corresponding to user activities. Given a probability p that user activity occurs independently in n time periods, AC can be estimated as the empirical

Table 3. Correlation and Traffic Overhead Comparison of Three Different Major Traffic Modification Approaches

	PCC	SRCC	Traffic Overhead (MB per Device per Day)
Pure Traffic Injection	0.75	0.83	29.11
Hybrid Traffic Reshaping	0.46	0.71	32.12
Random Traffic Padding	0.58	0.69	49.78
PrivacyGuard	0.35	0.64	27.15



(a) The accuracy comparison of occupancy detection (b) The amount comparison of additional traffic

Fig. 13. The comparison of occupancy detection accuracy and additional traffic after applying four different approaches.

fraction of n time periods with traffic corresponding to user activities; q is the probability decision function choosing to perform non-activity traffic padding. Thus, AC can be defined as

$$AC = \frac{np}{np + n(1-p)q}. \quad (5)$$

6.4 Experimental Results

6.4.1 Preventing Binary Occupancy Detection. We first compare the ability of four different approaches regarding masking occupancy. Note that our focus in this group of experiments is on binary occupancy status detection as opposed to the detection of multiple user activities classifications (as shown in Section 6.4.3). These approaches split the dataset into training and testing datasets using a ratio of 7:3 after cross validation. To ensure fair comparison, we set the traffic "cap" for each approach as 75 MB per device per day. As shown in Table 3, PTI receives PCC and SRCC values of 0.75 and 0.83, and HTR reports PCC and SRCC values of 0.46 and 0.71, respectively. RTP reports smaller PCC and SRCC values of 0.58 and 0.69, respectively, whereas PrivacyGuard yields the smallest PCC and SRCC values of 0.35 and 0.64, respectively. Thus, among all four different approaches, PrivacyGuard is the best-performing approach to hide user occupancy. As shown in Figure 13(a), we also compare occupancy detection accuracy when applying ML/DL-based attacks that we implemented in Section 3 to quantify the performance of HTR, RTP, and PrivacyGuard using the MCC. PrivacyGuard yields an average MCC of 0.2235, which is much closer to random detection (i.e., an MCC of 0.0) and a factor of more than two times less than the average MCC when attacking on HTR modified traffic, which is 0.4665.

Results. *By lowering the average MCC to 0.2235 in four smart homes, the PrivacyGuard approach effectively prevents occupancy detection from a wide set of ML-based and DL-based attacks. In addition, PrivacyGuard yields a factor of more than two times less than the MCC of occupancy attacks on the modified IoT traffic traces by prior approaches.*

6.4.2 Quantifying Traffic Overhead. We quantify the amount of network traffic overheads that are required to perform HTR, RTP, and PrivacyGuard. Figure 13(b) reports the amount of additional

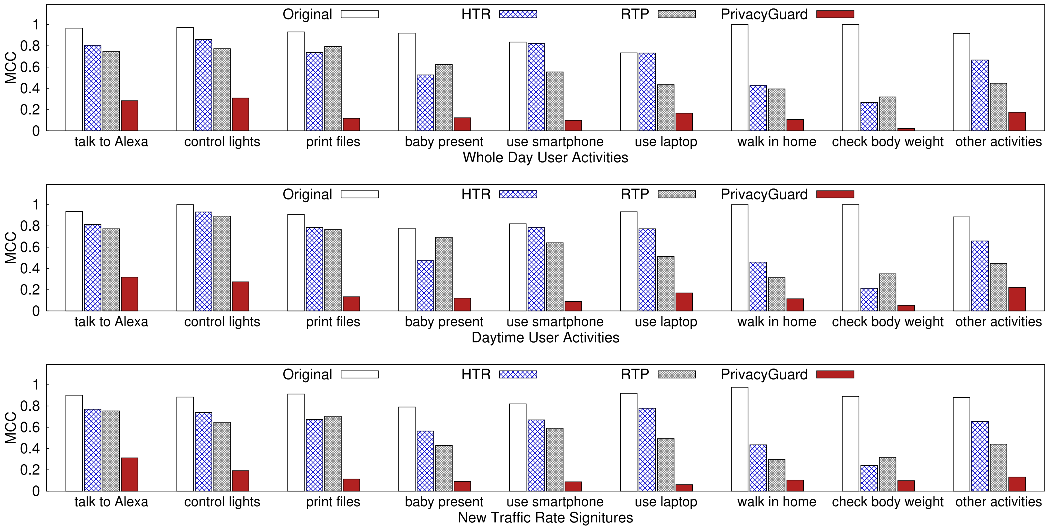


Fig. 14. The whole day (top), daytime (middle), and daytime with new traffic patterns (bottom) MCC comparison of user activities detection before (original) and after applying HTR, RTP, and PrivacyGuard.

traffic consumption for each approach. As expected, PrivacyGuard only consumes 27.15 MB of traffic per day on average over four datasets, which is about 2.7 times less than that of HTR, which is 73.11 MB on average per day. In other words, PrivacyGuard consumes the least amount of traffic overhead while achieving the best performance to prevent user privacy leakage in four smart homes.

Results. *PrivacyGuard only consumes 27.15 MB of traffic per day, which is about 2.75 times less than that of HTR, which is 73.11 MB per day. PrivacyGuard consumes the least amount of traffic overhead while achieving the best performance to prevent user privacy leakage.*

6.4.3 Preventing User Activities Detection Attacks. We next benchmark the effectiveness of masking user activities when applying three different privacy preserving approaches. We leverage ML/DL-based attack models that we built in Section 3 to detect nine different user activities using the original, HTR modified, and PrivacyGuard modified traffic rate traces. Unsurprisingly, as shown in Figure 14, PrivacyGuard always yields the worst MCC in both whole day (top) and midday (7 am to 12 am, bottom), and thus is the most effective privacy leakage preventing technique. In addition, we observe that the MCCs of both PrivacyGuard and PrivacyGuard using midday data only are the same or slight higher than the MCCs when using whole day data. This is mainly due to fact that users are typically sleeping at nighttime (12 am to 7 am), and thus most of the traffic occurrences in this period are not reflecting user in-home interactive activities. Therefore, all three approaches are reporting the same or slightly higher than the MCCs when attacking the midday (7 am to 12 am) traffic traces which have eliminated those "non-interactive" periods and mainly focus on user interaction patterns. Note that we observed the same trend when using both the UNSW dataset and the Smart* dataset.

We also examine the performance of the different defending approaches when handling new signatures. The goal is to benchmark these traffic reshaping approaches when the incoming traffic has a mix of known (pre-seeded) and unknown (new) IoT traffic rate signatures. For this example, we set the ratio to 1:1, and we find that all traffic reshaping approaches achieved MCCs similar to those shown in Figure 14 (middle) which primarily reshaped traffic rate traces using known

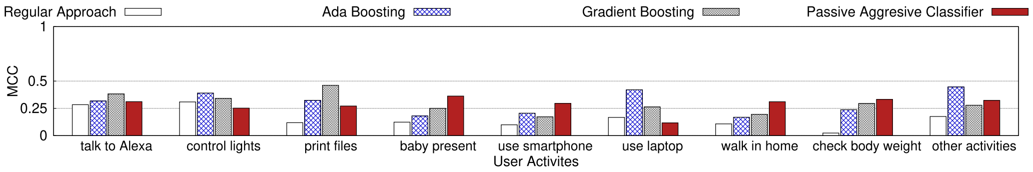


Fig. 15. The whole day MCC comparison of three complex attacking models targeting on PrivacyGuard.

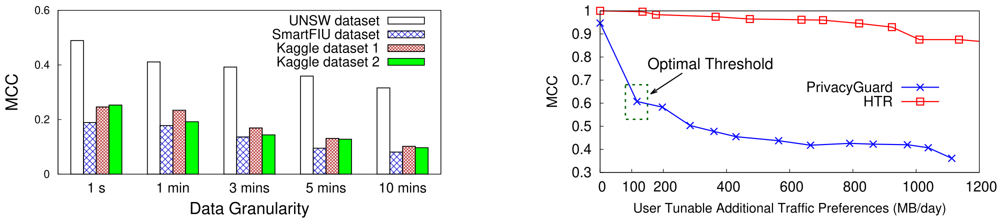
(pre-seeded) traffic rate signatures, and PrivacyGuard consistently yields the worst MCC. This is mainly due to the fact that the different reshaping approaches we implemented are handling the traffic rate signatures in the same manner, despite whether they are known or unknown signatures.

In addition to the preceding traditional single classifier ML or DL attacks, we examine the robustness of PrivacyGuard against complex and stronger attacks using the best-performing “ensemble” methods, such as Gradient Boosting, Ada Boosting, and Passive Aggressive Classifier, which could combine the “advantages” from multiple single ML and DL classifiers to stronger against PrivacyGuard. In doing so, we are examining the robustness performance of four user privacy defending approaches when they encountering smarter and more comprehensive attacks. As shown in Figure 15, we find that the MCCs of the four different approaches slightly increase due to the more powerful ensemble/complex attacks. However, the PrivacyGuard approach consistently achieves the lowest MCC results, with an upper bound of 0.46. PrivacyGuard is significantly more effective in defending against complex attacks compared to previous HTR or RTP defense methods.

We then examine the robustness bound by turning the user setting for their preference on preserving their data privacy. We find that PrivacyGuard, with automatically learned settings (resulting in 75% traffic reshaping), can achieve a bound MCC of 0.462, and with optimally user “tuned” settings, it can reach an MCC of 0.341 (with 75% artificial traffic), 0.273 (with 85% artificial traffic), and 0.202 (with 95% artificial traffic). Given the UNSW (with 20.5 days) dataset, with a mix of genuine user traffic (25%) and reshaped traffic (75%) with automatically learned settings, the robustness bound of PrivacyGuard is proportional to the volume of reshaped traffic in their home. However, for the scenarios where smart home users opt in their higher user privacy preserving, the robustness bound of PrivacyGuard is proportional to both the user preferred setting and volume of reshaped traffic. We find that we could fit these observations of attack accuracy and user tunable setting into polynomial regression function. Note that our main focus is to build systems to help users mitigate or reduce their data privacy leakage. There could be better potentially fitting models, which we will further explore in our near future work.

Results*. Our PrivacyGuard approach effectively prevents nine different user activities from a wide set of ML-based, DL-based, and ensemble sophisticated attacks in smart homes. Compared with prior approaches, PrivacyGuard consistently yields the lowest MCC for each user activity and thus is the best-performing privacy preserving approach.*

6.4.4 Quantifying Accuracy When Varying Granularity of Traffic Traces. We next evaluate the user activity detection effect on different traffic rate traces that have different levels of granularity, such as 1 second, 1 minute, 3 minutes, 5 minutes, and 10 minutes. By doing this, we can examine PrivacyGuard’s accuracy when attacking on different granularities of traffic traces. As shown in Figure 16(a), as expected, higher granularity results in lower user activity detecting accuracy in the MCC. This is mainly due to the facts that (1) PrivacyGuard performs consistently well on different traffic trace data at different granularities, and (2) fewer fluctuations and spikes are observed in higher-resolution traffic rate traces. In addition, when traffic rate traces are becoming coarser,



(a) Different Data Granularity

(b) Different User "Tunable" Additional Traffic Preferences

Fig. 16. The accuracy comparison of user activities detection when applying PrivacyGuard on traffic rate data in different granularities and different user "tunable" additional traffic preferences.

some principle features, such as standard deviation, variation coefficient, and AUC, become less distinguishable and thus are hidden. This will further obscure use activity information exposed in the traffic rate traces.

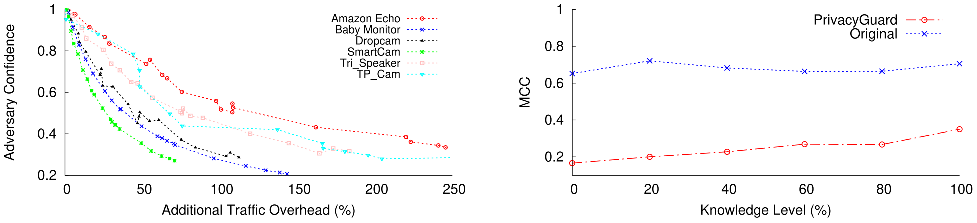
***Results**. PrivacyGuard's accuracy is a linear function of the granularities of IoT traffic rate traces. PrivacyGuard yields the MCC of 0.149 when attacking on 10 minute level network traffic rate traces, which is nearly the same as random prediction—that is, an MCC of 0.0.*

6.4.5 Quantifying Accuracy When Varying User Tunable Reshaping Preferences. We then evaluate the user activity detecting accuracy effect on different user tunable reshaping preferences. As discussed in Section 4.5, PrivacyGuard can be tuned by users based on their own preferences to achieve the balance between user privacy masking and additional traffic overhead. As shown in Figure 16(b), PrivacyGuard significantly reduces user activity detection accuracy—an MCC from ~0.95 to ~0.5—after applying the users' allowance of additional traffic as 6.45 MB per device per day (equivalent to ~1.83 KB per device per minute). However, the HTR approach only decreases the MCC from ~1.0 to ~0.96. In addition, under the overhead of 11.61 MB per device per day, PrivacyGuard yields an MCC of 0.47, which is two times less than HTR's MCC of 0.97.

***Results**. PrivacyGuard enables users the flexible control of threshold and artificial data injection to achieve a tradeoff between user privacy preserving and traffic overhead. In addition, when applying an additional 11.61 MB per device per day traffic overhead, PrivacyGuard yields an MCC of 0.46, which is two times less than the MCC of 0.97 using HTR.*

6.4.6 Preventing User Activities Detection by Adaptive Adversary. We next examine the effect of a different level adaptive adversary that has different AC on PrivacyGuard's user privacy enhancement performance. As shown in Figure 17(a), for the top-6 traffic-consuming IoT devices, the adversary's confidence diminishes significantly when PrivacyGuard has a higher allowance for traffic overhead. In particular, the cameras (e.g, baby monitor, drop camera, and smart camera) report the fastest AC decreasing. This is mainly due to the fact that these IoT cameras themselves have more "unstable" patterns compared to other devices, which enables them to more effectively adapt to the presence of our PrivacyGuard system. Figure 17(b) shows the ability of PrivacyGuard to preserve user privacy when an adaptive adversary has more prior knowledge about our deployed defense. In this group of experiments, we use the attacker knowledge level as the metric to represent the capabilities of different adaptive adversaries.

In essence, we use the percentage of traffic rate testing dataset that an external adversary poses to train the adversary ML or DL models to describe the attacker knowledge level to infer user in-home activities. A value of 0% indicates that an external adversary has no prior knowledge of the target home testing dataset and thus no cross validation is performed in their modeling,



(a) Adversary Confidence

(b) Adaptive Adversary

Fig. 17. PrivacyGuard performance comparison under different adaptive adversaries.

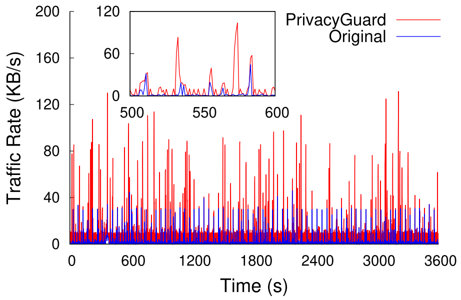


Fig. 18. Masking user activities with the PrivacyGuard prototype. The zoomed-in inset shows PrivacyGuard's online operations to hide user in-home traffic patterns.

whereas a value of 100% means that the external adversary has observed or recovered all prior knowledge about groundtruth traffic patterns for each user activity such that the attack models are "perfectly" trained and tested using the same testing dataset. The goal of this experiment is to understand our system's capability to protect user privacy under the attacks from the "adaptive" adversaries who have or reveal different traffic rate knowledge levels of the target smart home. We find that PrivacyGuard modified the traffic's MCC with slight increases from 0.16 to 0.35, whereas the original traffic's MCC fluctuates between 0.65 and 0.72. This is because attacking original traffic to infer user in-home activities is surprisingly easy, as we showed in Section 2 and Section 3. Note that even when an adversary has 100% knowledge about PrivacyGuard's DL-based defense model, PrivacyGuard still can prevent traffic analytics attacks at an MCC of 0.35, which is the almost two times less than the original traffic's MCC of 0.72.

***Results**. Using PrivacyGuard, the adversary's attack confidence significantly drops when a user permits additional overhead. In addition, PrivacyGuard yields an MCC of 0.35, which is almost two times less than that of original traffic when an adaptive adversary has 100% prior knowledge of our PrivacyGuard's modeling.*

6.4.7 Prototype Demonstration. Figure 18 demonstrates the performance of the PrivacyGuard prototype for a 3,600-second (1 hour) period of online traffic rate data (in kilobytes per second). The unmodified (original) traffic demand is the home's demand without PrivacyGuard's contribution. In contrast, the PrivacyGuard-modified demand is the external traffic rate trace seen by the adversaries, which includes using the prototype with low-cost artificial traffic signature injection, partial traffic reshaping, and online optimizations to mask private information exposed in traffic rate traces. The experiment shows how PrivacyGuard prototype modifies a home's traffic demand in real time, including both replaying artificial traffic rate signatures and partial

Table 4. IoT Device Response Time without and with PrivacyGuard

IoT Device	Original	With PrivacyGuard
Amazon Alexa	0.35s	0.51s
Google Home	0.42s	0.47s
Belkin Switch	0.47s	0.76s

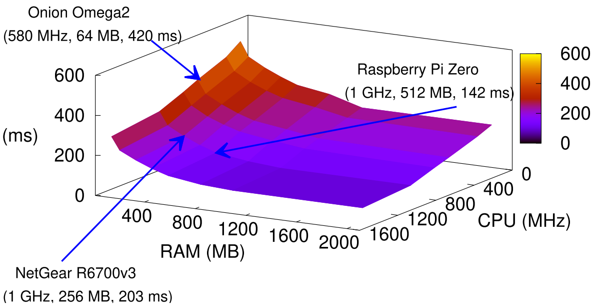


Fig. 19. The benchmarking results when varying memory size and CPU frequency of potential PrivacyGuard hosts.

traffic reshaping, to mask the traffic usage trends exposed in traffic rate traces. Our prototype demonstrates that PrivacyGuard's approach permits a straightforward implementation using widely used, off-the-shelf components. As shown in Table 4, PrivacyGuard can enable users to significantly reduce privacy leakage while still permitting regular IoT device usage.

Results. *PrivacyGuard functionality is simple to implement and deploy, requiring only the mechanism of basic hardware deployment and the ability to programmatically reshape traffic rates in real time.*

6.4.8 Hosting PrivacyGuard on Different IoT Devices. As shown in this work, PrivacyGuard can be deployed on Raspberry Pi or other IoT hubs in smart homes. In practice, many other already deployed IoT hubs, gateways, or devices potentially can be used to host PrivacyGuard. To evaluate this potential of the PrivacyGuard system, we further benchmark and examine the performance effect when PrivacyGuard is "deployed" on different memory size and different CPU frequency IoT devices. Figure 19 shows the benchmarking results of PrivacyGuard on different level computing resource IoT devices. To generate these results, we use Raspberry Pi to simulate different resource limited IoT devices by limiting its computing resources by using the Linux OS built-in tool `cpufreq-set`, which allows us to modify the settings of CPU and memory resources. For instance, `cpufreq-set -u 12MHz -d 12MHz` will reduce CPU frequency to 12 MHz. Similarly, we can grant more memory to the GPU and thus squeeze the RAM size available (as low as 2 MB) to the CPU. To benchmark the performance of different simulated IoT devices, we also prepared an IoT traffic benchmarking trace that comprises 1-hour traffic rate data generated by 22 IoT devices in 1-second granularity from Dataset #3 in Section 6.1.

As shown in Figure 19, the execution time (in the Z axis) that PrivacyGuard requires to reshape 1 hour long IoT traffic data decreases when the host's memory size is expanding. We observe that an IoT device with a CPU frequency of 60 MHz and a RAM size of around 15 MB can achieve a tradeoff of only around 550 ms execution time between PrivacyGuard performance and memory

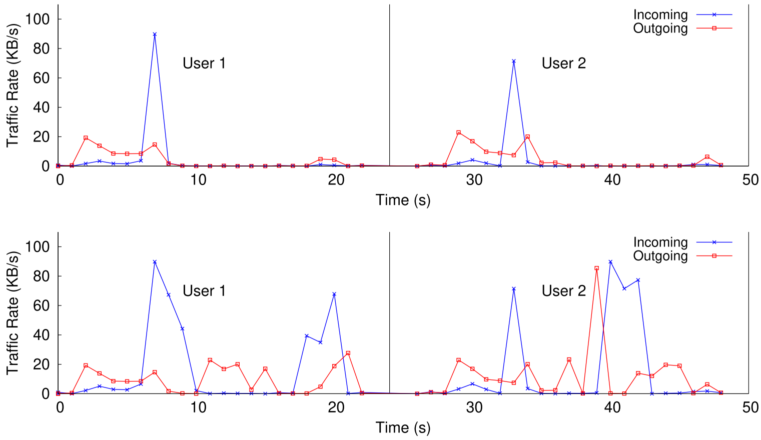


Fig. 20. Fingerprinting before and after applying PrivacyGuard using Amazon Alexa to query the weather conditions.

size. Similarly, we observe a similar performance trend when varying CPU frequency from 10 to 1,600 MHz in Figure 19. We find the tradeoff when we set the CPU frequency at 45 MHZ and a memory size of 15 MB. Thus, the benchmarking results verify the potential that PrivacyGuard can be deployed on many other IoT devices in smart homes. For instance, when deploying on the Arduino Mega 2560, which is typically equipped with a 16 MHZ ARM CPU and 8 KBytes of memory, PrivacyGuard can reshape 1 hour IoT traffic data within 1 second. We also pinpointed other potential hosts for PrivacyGuard in Figure 19, such as Onion Omega2 (CPU: 580 MHz; RAM: 64 MB; execution time: 420 ms), NETGEAR R6700 v3 (CPU: 1 GHz; RAM: 256 MB; execution time: 203 ms), and Raspberry Pi V0 (CPU: 1 GHz; RAM: 512 MB; execution time 142 ms). Note that evaluating the effectiveness of more smart home IoT devices will be further examined in our future work and is outside the scope of this work.

Results. *PrivacyGuard's performance improves when the host's computing resources are expanding. In particular, we observe that PrivacyGuard can achieve a tradeoff between computing resources and its privacy preserving ability. This further shows the potential to deploy PrivacyGuard on other already deployed IoT devices in smart homes.*

6.4.9 Preventing Traffic Rate Signature Fingerprinting. User-interactive IoT devices, especially for voice assistants (e.g., Amazon Echo and Google Nest Home), pose significant fingerprinting inference privacy threats when compared with other regular IoT sensors or switches. As shown in Figures 20 and 21, when three different smart home users are feeding the same voice commands to their voice speaker assistants, the generated traffic rate traces illustrate identifiable incoming/outgoing traffic rate signatures. In other words, it is very feasible to fingerprint user voice commands using only their engaged traffic rate traces. In addition, Table 5 further evaluates the similarity of the traffic rates generated by same voice commands before and after applying our PrivacyGuard approaches. In particular, the PCC and the SRCC have decreases on average of 0.304 and a maximum of 0.775. After applying PrivacyGuard, the traffic rate pattern generated by the same command shows a significant drop on the correlation coefficients by an average of 0.417 and 0.415, respectively.

Results. *PrivacyGuard can help smart home users significantly defend against user in-home activities fingerprinting inference attacks.*

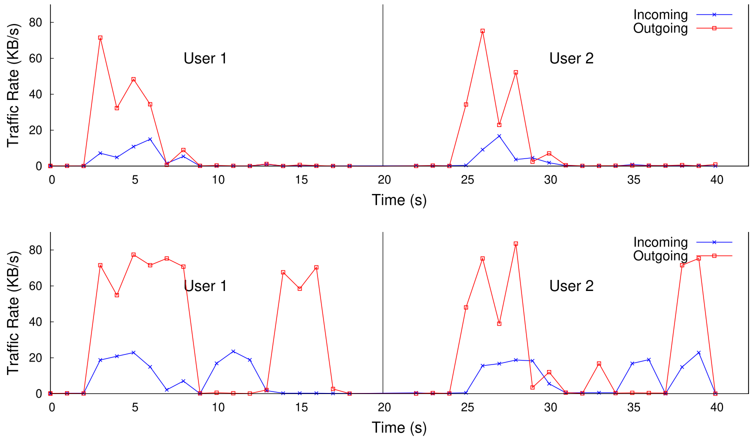


Fig. 21. Fingerprinting before and after applying PrivacyGuard using Google Nest Home to control the lights.

Table 5. Coefficient Comparison (in Terms of the PCC and SRCC) before and after Applying PrivacyGuard on the Aggregated Traffic Rates Generated by the Same Commands from Two Different Smart Home Users

Device		Before Applying PrivacyGuard	After Applying PrivacyGuard
Amazon Echo	PCC	0.657	0.082
	SRCC	0.746	0.112
Google Home	PCC	0.776	0.518
	SRCC	0.621	0.425

Table 6. Monthly Cost Comparison of PrivacyGuard and Six Other Recent Approaches

Approaches in USD	**Amazon AWS t2.nano**	**Google Cloud e2.micro**	**Microsoft Azure A0**
Pure Traffic Injection	3.15	8.29	16.08
Hybrid Traffic Reshaping	3.36	8.50	16.08
Random Traffic Padding	5.04	9.55	17.13
Tor	1.05	6.61	14.82
RepEL	3.36	8.29	16.08
Tamaraw	6.09	10.39	17.76
PrivacyGuard	2.97	8.14	15.96

6.4.10 System Cost Analytics. As discussed in prior sections, PrivacyGuard requires a remote cloud server to mimic bidirectional network traffic for some smart and user intensive interaction IoT devices. We examine the cost to run PrivacyGuard and other recent approaches to hide user private information in their network traffic traces. Table 6 shows the monthly household cost comparison results of seven different user privacy preserving approaches, including PrivacyGuard. To report the results in Table 6, we perform three online quote estimations from Amazon AWS (using t2.nano instances), Google Cloud (using e2.micro instances), and Microsoft Azure (using A0 instances), respectively. We assume that the targeted home is a standard one and has 21 popular IoT devices.

As shown in Table 6, PrivacyGuard only requires $2.97, $8.14, and $15.96 per month for 21 IoT devices when employing Amazon t2.nano, Google e2.micro, and Azure A0 to mimic IoT manufacturer remote servers, respectively. Note that although Tor [12] showed better monthly savings than PrivacyGuard, it is less effective when protecting user privacy leakage in network traffic data (shown in Figure 13 and Table 3). Therefore, PrivacyGuard achieves the best tradeoff between monthly cost and user privacy preserving.

Results. *PrivacyGuard only requires as low as $2.97 per month for a 21 IoT devices instrumented smart home to sufficiently protect user in-home private information. PrivacyGuard enables smart home users to achieve the balance between computing resources and their data privacy preserving.*

7 Conclusion and Future Work

We designed a new low-cost, open source user “tunable” defense system—PrivacyGuard—that enables users to significantly reduce the private information leaked through IoT device network traffic data, while still permitting sophisticated data analytics or control that is necessary in smart home management. We evaluated PrivacyGuard using IoT network traffic traces of 31 IoT devices from five smart homes and deploying a Raspberry Pi 4 based prototype. We found that PrivacyGuard enables smart home users to achieve the tradeoff between data utility and data privacy, and can effectively prevent a wide range of state-of-the-art ML-based and DL-based occupancy and another nine user activity detection attacks. We plan to collect more IoT traffic traces to further understand the tradeoff between privacy preserving and traffic overhead. We will also improve our user interface to promote user experience. Additionally, we plan to benchmark more hosts of PrivacyGuard on real IoT devices and develop a tailored smart router operating system that can host PrivacyGuard services directly.

Acknowledgment

We would like to thank the anonymous reviewers for their insightful comments, which significantly improved the quality of this article.

References

[1] Amazon. n.d. Differential Privacy. Retrieved October 25, 2024 from https://www.amazon.science/tag/differential-privacy

[2] Ostinato. 2020. Ostinato: Packet Generator and Network Traffic Generator. Retrieved October 25, 2024 from https://ostinato.org/

[3] GitHub. 2021. PrivacyGuard. Retrieved October 25, 2024 from https://github.com/cyber-physical-systems/PrivacyGuard

[4] Apple. 2024. Differential Privacy Overview. Retrieved October 25, 2024 from https://www.apple.com/privacy/docs/Differential$_P$rivacy$_O$verview.pdf

[5] Google Cloud. 2024. Use Differential Privacy. Retrieved October 25, 2024 from https://cloud.google.com/bigquery/docs/differential-privacy

[6] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16)*. 265–283.

[7] Josephine Akosa. n.d. *Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data*. Paper 942-2017. Oklahoma State University.

[8] Ahmed Alshehri, Jacob Granley, and Chuan Yue. 2020. Attacking and protecting tunneled traffic of smart home devices. In *Proceedings of the 10th ACM Conference on Data and Application Security and Privacy (CODASPY ’20)*. ACM, New York, NY, USA, 259–270. https://doi.org/10.1145/3374664.3375723

[9] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the smart home private with smart(er) IoT traffic shaping. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 128–148.

[10] Phuthipong Bovornkeeratiroj, Srinivasan Iyengar, Stephen Lee, David Irwin, and Prashant Shenoy. 2020. RepEL: A utility-preserving privacy system for IoT-based energy meters. In *Proceedings of the 2020 IEEE/ACM 5th International Conference on Internet-of-Things Design and Implementation (IoTDI '20)*. IEEE, 79–91.

[11] T. Brewster. 2017. Now Those Privacy Rules Are Gone, This Is How ISPs Will Actually Sell Your Personal Data. Retrieved October 25, 2024 from https://www.forbes.com/sites/thomasbrewster/2017/03/30/fcc-privacy-rules-how-isps-will-actually-sell-your-data/

[12] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, New York, NY, USA.

[13] Dong Chen, David Irwin, Prashant Shenoy, and Jeannie Albrecht. 2014. Combined heat and privacy: Preventing occupancy detection from smart meters. In *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communications*. 208–215.

[14] Albert Cheu, Adam Smith, and Jonathan Ullman. 2021. Manipulation attacks in local differential privacy. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP '21)*. IEEE, 883–900.

[15] Wikipedia. n.d. Cohen's Kappa. Retrieved October 25, 2024 from https://en.wikipedia.org/wiki/Cohen%27s$_k$appa

[16] Trisha Datta, Noah Apthorpe, and Nick Feamster. 2018. A developer-friendly library for smart home IoT privacy-preserving traffic obfuscation. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*. ACM, New York, NY, USA, 43–48.

[17] DD-WRT. n.d. DD-WRT: a Linux based Alternative OpenSource Firmware. Retrieved October 25, 2024 from https://dd-wrt.com/support/router-database/

[18] Wenbo Ding and Hongxin Hu. 2018. On the safety of IoT device physical interaction control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. 832–846.

[19] Cynthia Dwork. 2006. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*. 1–12.

[20] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology—EUROCRYPT 2006*. Lecture Notes in Computer Science, Vol. 4004. Springer, 486–503.

[21] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP '12)*. IEEE, 332–346.

[22] ExpressVPN. n.d. ExpressVPN. Retrieved October 25, 2024 from https://www.expressvpn.com/

[23] Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 1 (1997), 119–139.

[24] Ibbad Hafeez, Markku Antikainen, and Sasu Tarkoma. 2019. Protecting IoT-environments against traffic analysis attacks with traffic morphing. In *Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops '19)*. 196–201. https://doi.org/10.1109/PERCOMW.2019.8730787

[25] Md. Kamrul Hasan, Husne Ara Rubaiyeat, Yong-Koo Lee, and Sungyoung Lee. 2008. A reconfigurable HMM for activity recognition. In *Proceedings of the 2008 10th International Conference on Advanced Communication Technology*, Vol. 1. IEEE, 843–846.

[26] Ahmed Mohamed Hussain, Gabriele Oligeri, and Thiemo Voigt. 2021. The dark (and bright) side of IoT: Attacks and countermeasures for identifying smart home devices and services. In *Security, Privacy, and Anonymity in Computation, Communication, and Storage*. Lecture Notes in Computer Science, Vol. 12383. Springer, 122–136.

[27] Jiankai Jin, Eleanor McMurtry, Benjamin I. P. Rubinstein, and Olga Ohrimenko. 2022. Are we there yet? Timing and floating-point attacks on differential privacy systems. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP '22)*. IEEE, 473–488.

[28] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an efficient website fingerprinting defense. In *Proceedings of the European Symposium on Research in Computer Security*. 27–46.

[29] Sean Kennedy, Haipeng Li, Chenggang Wang, Hao Liu, Boyang Wang, and Wenhai Sun. 2019. I can hear your Alexa: Voice command fingerprinting on smart home speakers. In *Proceedings of the 2019 IEEE Conference on Communications and Network Security (CNS '19)*. IEEE, 232–240.

[30] Nikhil Ketkar. 2017. Introduction to Keras. In *Deep Learning with Python*. Springer, 97–111.

[31] Jinyang Li, Zhenyu Li, Gareth Tyson, and Gaogang Xie. 2020. Your privilege gives your privacy away: An analysis of a home security camera service. In *Proceedings of the 2020 IEEE Conference on Computer Communications (INFOCOM '20)*. IEEE.

[32] Nicole Lindsey. 2019. Smart Devices Leaking Data to Tech Giants Raises New IoT Privacy Issues. Retrieved October 25, 2024 from https://www.cpomagazine.com/data-privacy/smart-devices-leaking-data-to-tech-giants-raises-new-iot-privacy-issues/

[33] Jianqing Liu, Chi Zhang, and Yuguang Fang. 2018. Epic: A differential privacy framework to defend smart homes against Internet traffic analysis. *IEEE Internet of Things Journal* 5, 2 (2018), 1206–1217.

[34] Wikipedia. n.d. Matthews Correlation Coefficient. Retrieved October 25, 2024 from https://en.wikipedia.org/wiki/Matthews%$_c$orrelation%$_c$oefficient

[35] Mirimir. 2018. Collection of User Data by ISPs and Telecom Providers, and Sharing with Third Parties. Retrieved October 25, 2024 from https://www.ivpn.net/blog/collection-of-user-data-by-isps-and-telecom-providers-and-sharing-with-third-parties

[36] Jon Brodkin. 2019. ISPs Lied to Congress to Spread Confusion about Encrypted DNS, Mozilla Says. Retrieved October 25, 2024 from https://arstechnica.com/tech-policy/2019/11/isps-lied-to-congress-to-spread-confusion-about-encrypted-dns-mozilla-says/

[37] Rishab Nithyanand, Xiang Cai, and Rob Johnson. 2014. Glove: A bespoke website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, New York, NY, USA.

[38] Jorge Ortiz, Catherine Crawford, and Franck Le. 2019. DeviceMien: Network device behavior modeling for identifying unknown IoT devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation (IoTDI '19)*. ACM, New York, NY, USA, 106–117. https://doi.org/10.1145/3302505.3310073

[39] Homin Park, Can Basaran, Taejoon Park, and Sang Hyuk Son. 2014. Energy-efficient privacy protection for smart home environments using behavioral semantics. *Sensors* 14, 9 (2014), 16235–16257.

[40] Wikipedia. n.d. Pearson Correlation Coefficient. Retrieved October 25, 2024 from https://en.wikipedia.org/wiki/Pearson$_c$orrelation$_c$oefficient

[41] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. 2019. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 1 (2019), 209–237.

[42] Antônio J. Pinheiro, Paulo Freitas de Araujo-Filho, Jeandro de M. Bezerra, and Divanilson R. Campelo. 2021. Adaptive packet padding approach for smart home networks: A tradeoff between privacy and performance. *IEEE Internet of Things Journal* 8, 5 (2021), 3930–3938. https://doi.org/10.1109/JIOT.2020.3025988

[43] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[44] Vasanthan Raghavan, Greg Ver Steeg, Aram Galstyan, and Alexander G. Tartakovsky. 2013. Coupled hidden Markov models for user activity in social networks. In *Proceedings of the 2013 IEEE International Conference on Multimedia and Expo Workshops*.

[45] Vasanthan Raghavan, Greg Ver Steeg, Aram Galstyan, and Alexander G. Tartakovsky. 2014. Modeling temporal activity patterns in dynamic social networks. *IEEE Transactions on Computational Social Systems* 1, 1 (2014), 89–107.

[46] Karsten Rothmeier, Nicolas Pflanzl, Joschka Hüllmann, and Mike Preuss. 2021. Prediction of player churn and disengagement based on user activity data of a freemium online strategy game. *IEEE Transactions on Games* 13, 1 (2021), 78–88.

[47] Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of the European Symposium on Research in Computer Security*. 18–33.

[48] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2019. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing* 18, 8 (2019), 1745–1759.

[49] Wikipedia. n.d. Spearman's Rank Correlation Coefficient. Retrieved October 25, 2024 from https://en.wikipedia.org/wiki/Spearman%27s$_r$ank$_c$orrelation$_c$oefficient

[50] Statista. 2016. Internet of Things Connected Devices Installed base Worldwide from 2015 to 2025 (in Billions). Retrieved October 25, 2024 from https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[51] Mostafa Uddin, Tamer Nadeem, and Santosh Nukavarapu. 2019. Extreme SDN framework for IoT and mobile applications flexible privacy at the edge. In *Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom '19)*. 1–11. https://doi.org/10.1109/PERCOM.2019.8767413

[52] Karen Simonyan and Andrew Zisserman. n.d. Very Deep Convolutional Networks for Large-Scale Visual Recognition. Retrieved October 25, 2024 from https://www.robots.ox.ac.uk/~vgg/research/very$_d$eep/

[53] Chenggang Wang, Sean Kennedy, Haipeng Li, King Hudson, Gowtham Atluri, Xuetao Wei, Wenhai Sun, and Boyang Wang. 2020. Fingerprinting encrypted voice traffic on smart speakers with deep learning. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20)*. ACM, New York, NY, USA, 254–265. https://doi.org/10.1145/3395351.3399357

[54] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective attacks and provable defenses for website fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security '14)*. 143–157.

[55] Tao Wang and Ian Goldberg. 2016. On realistically attacking Tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies* 4 (2016), 21–36.

[56] Tao Wang and Ian Goldberg. 2017. Walkie-Talkie: An efficient defense against passive website fingerprinting attacks. In *Proceedings of the 26th USENIX Security Symposium*. 1375–1390.

[57] Wei Wang, Mehul Motani, and Vikram Srinivasan. 2008. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*. 323–332.

[58] Qiuyu Xiao, Michael K. Reiter, and Yinqian Zhang. 2015. Mitigating storage side channels using statistical privacy mechanisms. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1582–1594.

[59] Sijie Xiong, Anand D. Sarwate, and Narayan B. Mandayam. 2018. Defending against packet-size side-channel attacks in IoT networks. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '18)*. 2027–2031.

[60] Yongxiang Zhao, Baoxian Zhang, Cheng Li, and Changjia Chen. 2017. ON/OFF traffic shaping in the Internet: Motivation, challenges, and solutions. *IEEE Network* 31, 2 (2017), 48–57. https://doi.org/10.1109/MNET.2017.1500057NM

Received 30 March 2023; revised 1 June 2024; accepted 16 October 2024