PrivacyGuard: Enhancing Smart Home User Privacy

Keyang Yu, Qi Li, Dong Chen, Mohammad Rahman
Florida International University
Miami, Florida, USA
{kyu009,qli027,dochen,marahman}@fiu.edu

Shiqiang Wang
IBM Research
Yorktown Heights, New York, USA
wangshiq@us.ibm.com

ABSTRACT

The Internet of Things (IoT) devices have been increasingly deployed in smart homes and smart buildings to monitor and control their environments. The Internet traffic data produced by these IoT devices are collected by Internet Service Providers (ISPs) and IoT device manufacturers, and often shared with third-parties to maintain and enhance user services. Unfortunately, extensive recent research has shown that on-path adversaries can infer and fingerprint users' sensitive privacy information such as occupancy and user in-home activities by analyzing IoT network traffic traces. Most recent approaches that aim at defending against these malicious IoT traffic analytics can not sufficiently protect user privacy with reasonable traffic overhead. In particular, many approaches did not consider practical limitations, e.g., network bandwidth, maximum package injection rate or actual user in-home behavior in their design.

To address this problem, we design 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. In essence, our approach employs intelligent deep convolutional generative adversarial networks (DCGANs)-based IoT device traffic signature learning, long short-term memory (LSTM)-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 5 smart homes. We find that PrivacyGuard can effectively prevent a wide range of state-of-the-art machine learning-based and deep learning-based occupancy and other 9 user in-home activity detection attacks. We release the source code and datasets of PrivacyGuard to IoT research community.

CCS CONCEPTS

• **Security and privacy** → *Social network security and privacy*; • **Computing methodologies** → *Machine learning approaches*; *Neural networks*; *Classification and regression trees.*

IPSN '21, May 18–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8098-0/21/05…$15.00
https://doi.org/10.1145/3412382.3458257

KEYWORDS

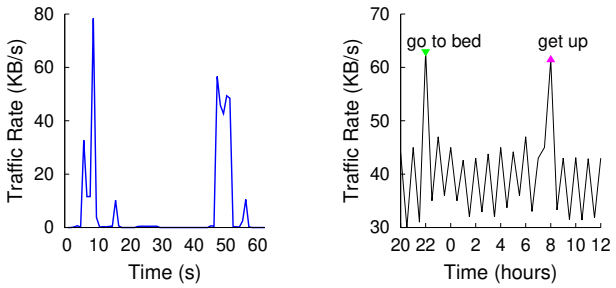
IoT privacy, Smart Home, Machine Learning, Deep Learning

ACM Reference Format:
Keyang Yu, Qi Li, Dong Chen, Mohammad Rahman and Shiqiang Wang. 2021. PrivacyGuard: Enhancing Smart Home User Privacy. In *The 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021) (IPSN '21), May 18–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3412382.3458257

1 INTRODUCTION

People are increasingly deploying the Internet of Things (IoT) devices in smart homes and smart buildings to monitor and control their environment. The total installed base of the IoT devices is projected to amount to 75.44 billions worldwide by 2025, a fivefold increase in 10 years [35]. This penetration of IoT devices holds great promise to transform people's lives by making society more efficient in many areas, including smart home, transportation, manufacturing, e-health, etc. Traffic data generated by these IoT devices is recorded by Internet Service Providers (ISPs) to maintain customer services, such as generating monthly bills, personalizing 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." [22]. 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 [23]. ISPs like AT&T, Comcast, Time Warner, Sprint, and Verizon are selling personal network traffic data without prior user consent to "enhance" user experience [7]. Also, recent IoT privacy study [19] shows that 72 out of 81 popular IoT devices are sharing data with third-parities (e.g., Google, Amazon and Akamai) completely unrelated to original manufacturer and far beyond basic necessary device configuration, including voice speakers, smart TVs, and streaming dongles.

In parallel, significant recent research [5, 6, 8, 9, 12, 13, 15, 25, 32, 39, 40] has shown that launching attacks to extract user activities from IoT traffic data is surprisingly easy, since user activities highly correlate with simple time-series data statistical metrics, such as mean, variance, and range. Thus, IoT device traffic data has significant privacy threats. 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 [18]. The network traffic rate trace (in kB/s) of 2 IoT devices from a single apartment is reported in Figure 1. The traffic rate trace signals the occupancy status in this home. Most recent research [16] also demonstrated that a passive Amazon Alexa attacker can fingerprint users' voice



(a) Voice Assistants (b) Sleep Sensor

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

commands and compromise the user privacy of millions of U.S. consumers. In addition, this 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 users are regularly having meals at home. Another example, consistent amount of TV network traffic on Saturday night from 7 pm to 9 pm may indicate the residents are watching NBA games every weekend. Also, a lack of significant network traffic may show that occupants are out of town for vacation. Intuitively, users' interaction with IoT devices, e.g., talking to voice assistants, opening/closing doors, watching smart TVs, lends itself to straightforward attacks that detect changes in these metrics and associates them with changes in user activities.

Prior research [5, 8, 11, 24, 37–39, 41] 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; (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 3~4 times additional traffic overhead. Thus, new lost-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 smart home 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. This paper makes the following contributions.

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 network traffic rate over time for popular IoT devices. In doing so, we review, implement, and benchmark a wide range of sophisticated user activities attack models using machine learning (ML) and deep learning (DL) approaches, including k-nearest neighbors (k-NNs), Hidden Markov Models (HMMs), Support Vector Machines (SVMs) and Convolutional Neural Network (CNNs).

PrivacyGuard Design. We present the design of PrivacyGuard, which enables users the "tunable" control to significantly reduce, the private information leaked through IoT device network traffic. In essence, PrivacyGuard employs intelligent deep convolutional generative adversarial networks (DCGANs)-based traffic signature learning, long short-term memory (LSTM)-based artificial traffic signature injection, and partial traffic reshaping to obfuscate user in-home 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 widely-used open-source frameworks. We evaluate PrivacyGuard using traffic rate traces of 31 IoT devices from 5 smart homes. The results show that PrivacyGuard effectively prevents 9 different state-of-the-art ML/DL-based user activity attacks. We evaluate PrivacyGuard in multiple ways: (1) We evaluate PrivacyGuard's defending performance using traffic traces of 25 IoT devices for 20.5 days from USNW dataset [33]. (2) We then evaluate PrivacyGuard's accuracy using traffic traces of 31 IoT devices in our own on-campus "mock" smart homes. (3) We validate PrivacyGuard's accuracy for using IoT traffic rate traces of 22 IoT devices from two townhouse apartments. (4) We also download two Kaggle datasets to evaluate PrivacyGuard.

Releasing Datasets and Code. Our new approaches to analyze IoT network traffic traces and prevent user sensitive information leakage in these traces using machine learning-based and deep learning-based traffic reshaping techniques are quite general, and can be applied to address similar privacy problems in other data analytics research domains, such as smart grid and medical e-health system. For instance, our feature selection approach from time-series data motifs can be applied to extract spike features in smart grid energy meter reading traces and healthcare device traffic traces. Our LSTM-based traffic reshaping approaches can also applied to preserve user sensitives information (e.g., occupancy, location, and daily routine) exposed in the time-series traces of smart meters and medical devices. We release the source code, datasets, and attack models that we used to design and evaluate PrivacyGuard to research communities on our website [2].

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 etc., are accessible to many 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 the external adversaries can use any data analytics techniques, such as data mining, ML/DL, inference, or other statistical methods to infer certain types of information for the observed patterns in the recorded traffic traces. Thus, inferring or discovering user activities in these homes is considered as an opposition to the users' privacy preferences.

In particular, we are concerned with 3 types of privacy attacks: i) *Learning occupancy from the data.* This includes whether a home is occupied and when; ii) *Learning user in-home activities from the traffic data.* User activities may include when users come and

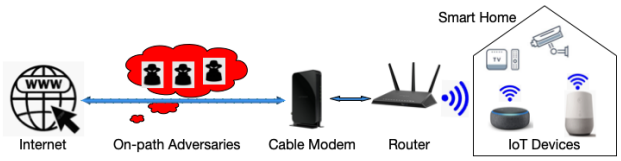


Figure 2: Overview of our privacy threat model.

go, when they perform their daily activities, such as going to bed, waking up, watching TV, listening to music, 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; iii) *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.

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 the real-time IoT network traffic traces and leverage ML/DL-based statistical learning and data mining approaches. Then, the external attacker may launch cyberattacks for 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 analytic approaches to learn the indirect user privacy information that might be interesting for insurance companies, marketers, or the government.

In addition, we assume 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 cloud servers is outside the scope of this paper.

2.2 Related Work

We outline the design alternatives to preserve smart home user privacy using pure traffic injection approach, hybrid traffic reshaping approach, and random traffic padding approach. In doing so, we review a wide range of the most recent sophisticated traffic reshaping-based prevention techniques [5, 6, 8, 9, 13–15, 25, 29–32, 39, 40] to thwart privacy attacks on IoT traffic rate traces.

To understand the performance of the above existing approaches, we implemented three different traffic reshaping approaches. Figure 3 shows a 3-day traffic rate (kB/min) traces comparison results using the three approaches. Table 1 quantifies the effectiveness of the three approaches and additional three recent approaches by showing Pearson Correlation Coefficient (PCC) and Spearman's Rank Correlation Coefficient (SRCC). The PCC [26] 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 [34] 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

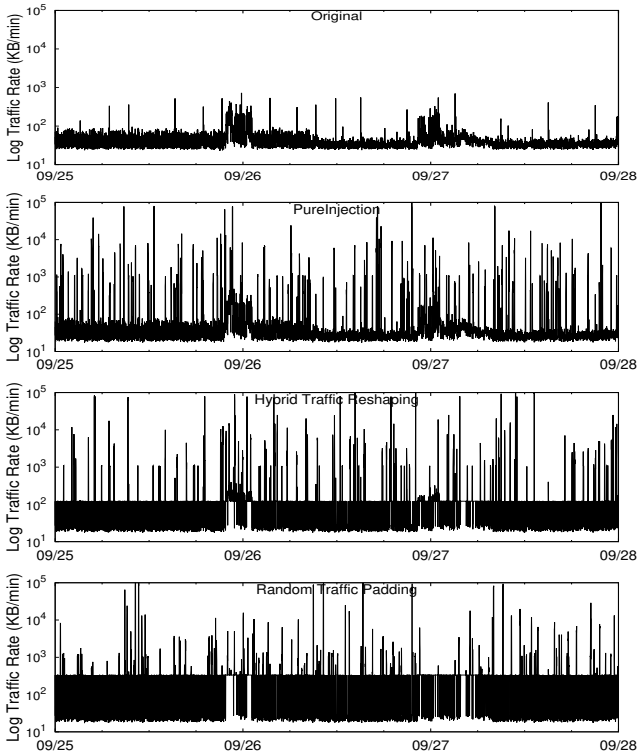


Figure 3: A 3-day traffic rate comparison of before and after applying 3 different major traffic reshaping approaches.

the original traffic rate traces. We use PCC and SRCC to quantify the effectiveness of the prior approaches on masking user private information. We also use ϵ-security [25] to describe the probability of a traffic reshaping approach can not prevent smart home users from an external adversary's in-home activity inferring.

Pure Traffic Injection. Prior work [8, 20, 25] proposed defense approaches to inject "fake" traffic patterns to conceal genuine user in-home network traffic patterns. As shown in Table 1, the general implementation yields additional overhead as ~97%. In particular, Park et al. found that traffic data encryption cannot prevent privacy invasions exploiting traffic pattern analysis and statistical inference [25]. Park et al. 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. presented a defense against Tor website fingerprinting that can reshape traffic rate traces by controlling the size of the parameter to pad packets [8]. However, these approaches did not completely hide the genuine network traffic patterns, in particular, during higher and lower traffic rates periods. This may still allow adversaries to distinguish "fake" traffic patterns from genuine traffic patterns to infer user in-home activities.

Hybrid Traffic Reshaping. Prior work [6, 9] presented hybrid reshaping techniques to prevent user privacy leakage in the aggregated sensor or 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

	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 [8]	0.805	0.712	77.5%	25%
RepEL [6]	0.361	0.525	33%	100%
Tamarow [8]	0.292	0.473	3.4%	199%

Table 1: The correlation and traffic overhead comparison of six major traffic modification approaches.

additional overhead as ~103.7%. Chen et al. proposed to learn the "noise" injection rate using empirical statistical analytics (e.g., probability mass function) of smart home device events [9]. Similarly, Bovornkeeratiroj et al. proposed RepEL which employed an edge gateway to partially flatten loads and randomly replay loads to hide private user occupancy information [6]. Shmatikov et al. 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 [32]. Wang at al. [40] designed a traffic padding algorithm that uses matched package schedules to prevent adversaries from paring incoming and outgoing traffic flows. Significant work [14, 29–31] 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 the random injecting of IoT traffic signatures, these approaches may still allow sophisticated attackers to identify the randomly injected "fake" signatures, and thus infer the genuine user private information.

Random Traffic Padding. Recent work [5, 13, 15] proposed random traffic padding approaches, that aims 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 as ~165.9%. Dyer et al. proposed a buffered fixed-length obfuscator based on random padding to prevent website fingerprinting attacks [13]. Juarez et al. proposed an adaptive padding approach that can provide a sufficient level of security against website fingerprinting [15]. The proposed approach matched the gaps between traffic packets with a distribution of generic network traffic. When a large gap is identified in the network traffic volume data, this approach will inject padding traffic in that gap to prevent long gaps from being a distinguishing feature for attackers. Similar to hybrid reshaping, Apthorpe et al. 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 [5]. Rather than using pre-defined IoT device traffic pattern distribution, Apthorpe et al. integrated their approach with Hidden Markov Model (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 shows that random traffic padding approach—Tamarow yields the lowest PCC, SRCC, and ϵ-security as of 0.292, 0.473, and 3.4%, respectively. Unsurprisingly, pure traffic injection approach reports the highest PCC, SRCC, and ϵ-security as of 0.75, 0.83, and 87.15%. This is mainly due to the fact that pure traffic injection 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. Hybrid traffic reshaping approach reports coarser correlation than pure traffic injection approach. This is because in addition to injecting "noise" into IoT traffic traces, hybrid traffic reshaping approach also makes its best efforts to partially flatten both genuine and "fake" traffic patterns of IoT devices. The different correlation performance between hybrid traffic reshaping approach and random traffic padding approach is due to the fact that random traffic padding approach generally has higher flattening threshold to pad IoT traffic patterns, and also consider the bidirectional traffic padding for IoT devices (e.g., Amazon Alexa, Google Home). For the same reason, random traffic padding approach—Tamarow reports the maximum traffic overhead as of 199% additional overhead per device per day. However, even the best performing approach—random traffic padding approach still reports significant value of PCC and SRCC. This is mainly due to fact that this approach may not consider practical limitations in real smart homes, such as 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 "spike" 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); (3) Flattening and injection operations have practical limitations, such as network bandwidth, maximum package injection rate, and user behavior permitting. And these may still allow adversaries to infer user in-home sensitive information by applying time-series data analytics attacks. In addition, the native flattening algorithms broadly employed by many approaches actually resulted in 3~4 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 we discussed in Section 2, we are concerned with user private sensitive information that may be learned by adversaries from the externally observed traffic rate traces of the IoT devices in smart homes. To explore the severity and extent of this privacy threat, we design a wide range of Machine Learning (ML)-based and Deep Learning (DL)-based user activities attack models to better understand and identify the most common user activities can be learned by those adversaries. Unlike the existing work mainly focusing on binary occupancy status detection, we investigate the multiple-class user activities when a home is occupied. In doing so, we identify the privacy leakage in the IoT network traffic rate traces. In addition, we use all these attack models that are developed in this section to evaluate our new approach—PrivacyGuard in Section 6.

To benchmark the performance for each attack model as shown in Table 2, we use the Matthews Correlation Coefficient (MCC) [21],

User Activities	Model	MCC	Cohen's Kappa
Talk to Alexa	Logistic Regression	0.966	0.983
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	Decision Tree	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	Decision Tree	0.889	0.999
Make Coffee	SVMs (Linear)	0.969	0.971
Other Activities	LSTM	0.917	0.927

Table 2: The best performing attack models to detect 13 different user activities using two datasets.

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 user activities detection is always wrong. The Cohen's Kappa [10] is a measure of the agreement between two classifiers who each classify N items into C mutually exclusive categories. The Cohen's kappa is widely used to evaluate multi-class classifiers. 1.0 indicates a complete agreement, while, κ=0 indicates no agreement among the multi-class classifiers. We will discuss more detail about MCC and Cohen's Kappa in Section 6.

3.1 Feature Selection

To identify principle features of IoT traffic rate data that we use to build and train our attack models, we first build a large IoT aggregated 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 each IoT device, including duration, mean, maximum and minimum values, standard deviations, range, Skewness, variation coefficient, kurtosis, area under the curve (AUC), etc. We leverage Principal Component Analysis (PCA) to analyze the principle features from IoT network traffic rate traces. As shown in Table 3, we find 8 principle statistical learning features to identify user activities-talking to voice assistant and taking care of baby to train our attack models. As show in Table 3, the features of the two user activities have significantly different values that allow our attack models to distinguish different user activities.

Many smart home IoT devices generate bidirectional traffic when occupants are interactively using them. For instance, a typical traffic rate trace of Amazon Echo may present a short burst of outgoing traffic and then multiple incoming traffic flows. Insteon Dimmer and Philips Hue Light show sparse bursts in both incoming and outgoing traffic traces when users are interactively controlling their indoor lights. Wink Door Sensor, Wemo Smart Switch, and Wemo Mini Plug typically generate very short and sparse and discrete "on" and "off" traffic pattern sister pairs. Baby Monitor and Drop Camera show the combination of periodic traffic spikes and sparse bursts of sharp traffic spikes in their traffic traces. In addition, different user activities typically have different length of duration which may further affect the feature extraction accuracy.

Features	Amazon Echo		Take Care of Baby	
	IN	OUT	IN	OUT
Total Traffic	0.000491	0.000399	0.001994	0.001473
Duration	0.142857	0	0.714285	1
Range	0.000490	0.000343	0.001826	0.003503
Mean	0.000142	0.000299	0.001647	0.006664
Standard Deviation	0.000448	0.000321	0.001735	0.003245
AUC	0.000119	0	0.001332	0.005062
Skewness	0.969411	0.969412	0.556685	0.556685
Variation Coefficient	0.730516	0.452877	0.262926	0.318911

Table 3: The selected features for two user activities.

To learn the sequential event characteristics (e.g, Standard Deviation, Skewness, and Variation Coefficient) exposed in IoT traffic rate traces, we leverage the sliding window-based feature extraction approach to further process the traffic spikes.

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 above-mentioned statistical time-series metrics using a sliding window n. To ensure the effectiveness of all the attack models on different user activities, we need to find the optimal sliding window n that can accommodate all the IoT devices in a smart home. Figure 4 shows the derivative of 2-degree polynomial fitting on our attack model performance (in MCC) when using different sliding window size—n. As shown in Figure 4, the sliding window size—n 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 attack models can observe and extract the principle features exposed in IoT traffic rate spikes to indicate user in-home activities.

When IoT traffic traces are aggregated in smart homes, several features (e.g, range, mean, standard deviation, and AUC) may present slight deviations from non-aggregated traffic feature results. However, we observe that among 72,370 traffic spikes, only 15.19% have significant aggregation due to the overlapped or interleaved IoT device usages in 34-day traffic traces from 4 smart homes. In addition, our attack models are trained on the most significant times-series motif features that are extracted on a significant amount of traffic spikes. These motifs retain the most significant patterns and have the least effect from the traffic aggregation. There is a potential limitation on the performance of our attack models on non-residential buildings due to a significant higher traffic aggregation level. For instance, in a smart commercial building that has a large number of occupants are interactively using IoT devices, the features that our attack models are relying on to infer user activity may be further obscured. A full evaluation of this effect is outside the scope of this paper.

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

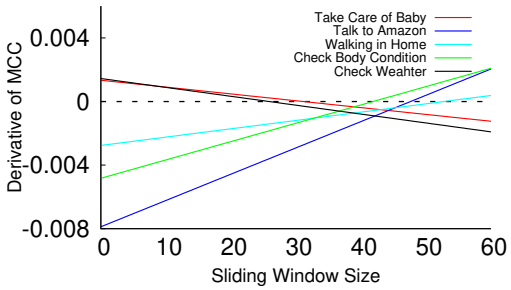


Figure 4: The relationship between sliding window and user activity inference accuracy using 31 IoT devices.

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 on our system's performance is discussed in Section 6.4.4.

3.2 Machine Learning-based 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, Support Vector Machines (SVMs), and Random Forest. In particular, we also benchmark different kernels for SVMs, including linear, linear passive-aggressive, linear ridge, polynomial with 1~10 degrees, and radial basis function (RBF). Table 2 shows the results for attacking 13 different user activities. Note that, for each user activity shown in Table 2, we run all the ML-based attack models, and report the one that has the best attacking accuracy in MCC and Cohen's Kappa.

3.3 Deep Learning-based Attacks

In addition the ML-based attack models, we also design a convolutional neural networks (CNNs)-based deep learning approach to detect user in-home activities from IoT traffic rate traces. Below, we describe the design of our CNNs architecture which is inspired by the most notable prior CNNs research—VGGnet [36]. As shown in Figure 5, our CNNs architecture is comprised 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 outputs.

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 IoT network traffic rate traces in a smart home. On average, our ML-based and DL-based attacking approaches yield the average MCC as 0.952 and the average Cohen's Kappa as 0.974. This results show that our implemented ML-based and DL-based attack approaches are such effective at detecting user's private sensitive information (e.g., user activities) in a smart home. Thus, IoT traffic rate traces expose a serious threat to user in-home privacy. Therefore, new privacy preserving techniques are necessary. We employ all the above attack models to evaluate our new approach—PrivacyGuard in Section 6.

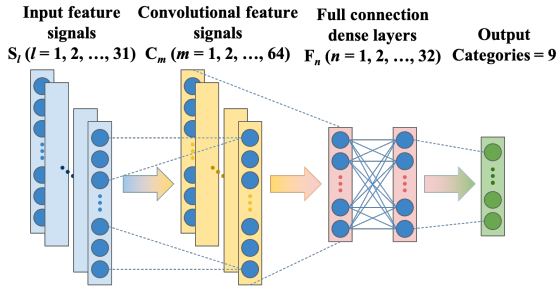


Figure 5: The overview of our CNNs architecture.

4 PRIVACYGUARD DESIGN

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

4.1 Privacy Guarantee

Differential privacy is a system that can ensure if an arbitrary single substitution in the IoT traffic rate traces is small enough, the statistical query learning results can not 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.

We employ (ϵ)-differential privacy as a formulated metric to describe the privacy guarantee of PrivacyGuard. The definition for (ϵ)-differential privacy is: *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 $S \subseteq Range(A)$:*

$$P[A(T_1) \in S] \leq P[A(T_2) \in S] \cdot exp(\epsilon)$$

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 is processed by applying the algorithm A and thus to predict the associated user in-home activity. The parameter ϵ is a quantitative measurement of the strength of our privacy guarantee. Lower values indicate a stronger guarantee in our system. However, we are not only focusing on the specific levels of differential privacy, but also are interested in understanding the trade-off between the complexity of the algorithm A (e.g., overhead) and user privacy.

Our goal is to develop a perfect algorithm A that can ensure if an arbitrary single substitution in the IoT traffic rate traces is small enough, the adversaries can not infer accurate user in-home sensitive information. PrivacyGuard preserves user differential privacy by combining intelligent traffic rate signature learning, artificial traffic rate signature injections, and partial traffic reshaping to approximate the algorithm A.

4.2 System Design

Figure 6 shows the system structure of our PrivacyGuard. PrivacyGuard assumes either a software virtual private network (VPN) or hardware VPN router is deployed in a smart home. PrivacyGuard

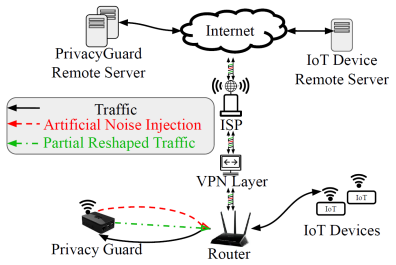


Figure 6: System model of PrivacyGuard.

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 6), home router, or home gateway. A VPN wraps all smart home traffic from IoT devices in an additional transport layer. By doing so, the VPN can aggregate all the 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 [5]. Then, PrivacyGuard takes additional steps 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 also employs multiple optimization techniques to further obscure the privacy information that are exposed in the externally observed traffic rate traces with lower traffic overhead. Figure 7 shows the 3 major operation flows of PrivacyGuard.

4.3 Intelligent Traffic Rate Signature Learning

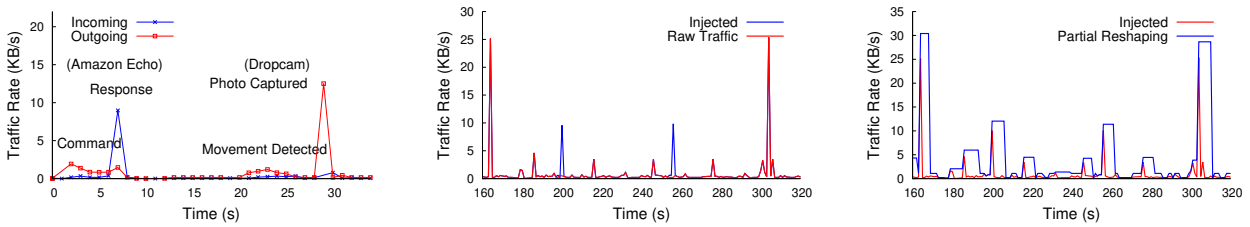
PrivacyGuard first learns IoT device traffic rate signatures that are used in its later traffic reshaping algorithms. The goal of this traffic rate signature learning is to ensure that it is reliably difficult for the external adversaries to distinguish the genuine IoT traffic rate signatures from the "artificial" 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 7 (a) shows the traffic rate signatures (in KB/s) of Dropcam and Amazon Alexa. We store all the traffic signatures for IoT devices in a SQLite database. PrivacyGuard also takes additional steps to ensure 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, e.g., short, long, high, low, medium, and 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 Pearson Correlation Coefficient (PCC) [26], which is a measure of the linear correlation between 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 1.

In addition, we also observed that some IoT devices, such as 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 Deep Convolutional Generative Adversarial Networks (DCGANs) [28] to build a new IoT traffic rate signature generator. Our DCGANs architecture is comprised 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, following this layer, we have five convolutional layers. For generator model, we use the ReLU activation function for all the layers except the final one, where we employ the Tanh activation function. Our generator and discriminator have almost the same architectures, but reflected. For discriminator model, we use the Leaky ReLU activation function for all the 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 device. 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 that is 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. And 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 the injected traffic patterns still fit the traffic distributions that represent the regular user in-home behaviors such that the external adversaries cannot distinguish the injected traffic patterns from the genuine traffic patterns. Next, we will explain how PrivacyGuard models user in-home activities.



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

Figure 7: PrivacyGuard's traffic signature learning (a), artificial traffic signature injection (b), and partial traffic reshaping (c).

Algorithm 1: 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 ∀ V_i ∈ 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 ∀ T_i ∈ T do
    for Day_i ∈ [0, 6] do
        for Hour_i ∈ [0, 23] do
            TC_i = TC_i + 1 // Update traffic frequency
            TV_i+ = TV_i    // Update traffic rate
```

Long short-term memory (LSTM)-based user in-home activities modeling. To address this problem, we present a recurrent neural network (RNN)-based approach to model real user in-home behaviors. Specifically, we design a LSTM-based approach to model user in-home behavior using IoT traffic rate traces. Note that, similar to HMM, LSTM-based approach also assumes user activities behind these IoT device events are hidden and thus can be learned though the LSTM architecture. Compared with 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 the IoT device, and the IoT devices reported sensor data is associated with 9 different user in-home activities. The output of the LSTM model is the future user activities. As shown in Figure 8, the first visible layer is LSTM layer with 10x10 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 the 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 Adam algorithm as the optimizer for our LSTM model that can better handle high dimensional parameters and mitigate sparse gradients 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 routine, user population, and user patterns may be different in different homes. In addition, in a new home, user activity, home configuration, and IoT devices may also vary. Users can deploy our PrivacyGuard to automatically retrain the above-mentioned LSTM model to learn these user in-home patterns which we had benchmarked in Table 3.

Bidirectional Traffic Signature Injection. The way that PrivacyGuard leverages to mimic unidirectional communication IoT devices is trivial. However, there are a significant amount of IoT devices are user interaction intensive, such as voice assists, IoT smart plugins, etc. and they have bidirectional traffic flows. To mimic these IoT devices, as shown in Figure 6, PrivacyGuard may be deployed both locally and on the remote servers using Master/slave model. The local PrivacyGuard works regularly as the master which is very similar to other single directional traffic IoT devices, while, 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. That says, 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 as the "valid" IoT remote server to respond to artificial IoT device bidirectional traffic demands. To mimic the incoming/inbound traffic, we build PrivacyGuard remote server on top of the traffic and package editor/generator—Ostinato [1] 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 Ostinato Python API [1] to vary packet fields

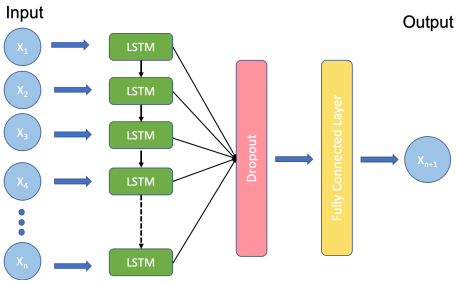


Figure 8: Overview of LSTM-based user activity modeling.

across packets at run time, e.g. changing the source IP/MAC addresses in the packages of PrivacyGuard remote server to the actual IoT remote server's (shown in Figure 9). 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 to better hide user sensitive information in the traffic rate traces.

Algorithm 2: 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 ∈ [0,23] do
        if TC_i ≥ 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 V̄_i from DB based on our learned user activity H
            /* Further obscure privacy in the load */
            Update traffic volume V_i = V_i + V̄_i
```

4.5 User Tunable 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 [5, 8, 11, 24, 37–39, 41], simply assuming 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 traffic demand to a target less than the peak traffic demand. $T_{current}(t)$, $U(t)$, $T_{average}(t)$ denotes the current traffic rate demand, user preferred set-point, and the average traffic

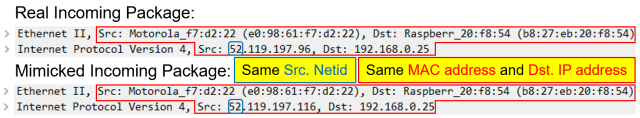


Figure 9: The illustration of PrivacyGuard remote server modified packages for Amazon Echo.

rate demand, respectively. In order 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 3.

Note that, PrivacyGuard can automatically learn an optimal/default trade-off 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.

Algorithm 3: 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+δ) = {V_t, V_{t+1}, ..., V_{t+δ}} where Σ_{t=1}^{V_t} := V
for ∀ V(t, t+δ) ∈ V do
    Set Reshaping_threshold = max{TV_t, U, Current_load_t}
    if V_t ≳ Reshaping_threshold then
        Continue
    else
        /* Reshaping traffic load using selected threshold */
        V_t = Reshaping_threshold
        Extend the reshaping for random ϵ seconds
        V(t, t+ϵ) = Reshaping_threshold
```

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%∼5% of traffic demand. In addition, for each traffic rate demand reshaping, PrivacyGuard also extends its duration by a small random amount, e.g., 0%∼5% of the

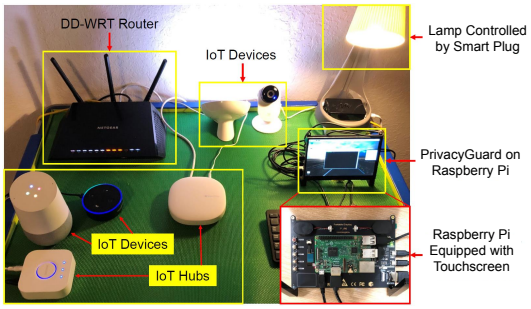


Figure 10: The overview of our PrivacyGuard prototype.

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, in night time when most of smart home users are sleeping, PrivacyGuard needs to ensure traffic traces have significant less interactive IoT device traffic demands.

Traffic Injection Rate Adjustment. Finally, 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 there is no need to make lower-traffic nighttime periods look like high-demand traffic daytime periods. Instead, PrivacyGuard only ensures these time periods look the same with respect to each other, regardless of whether a home is occupied or not. In addition, PrivacyGuard also 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.

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 10 and Table 4, we employ a Raspberry Pi 4 Model B-based hardware (Broadcom BCM2711, Arm Cortex-A72 Architecture) setup, which enable 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 Amazon EC2 t1.micro instance with a cost of $0.0035 per hour. We also stores the set of artificial traffic rate signatures, indexed by time period, that are available for replay in a *SQLite*3 database. The size of the implementation is less than 1500 lines of code. We use the Scikit-learn machine learning library in `python` to build our machine learning attack approaches. The library supports multiple techniques including Logistic Regression, Support Vector Machines (SVMs), and Random Forest. In particular, we also

	Metrics	Specifications
LSTM	CPU usage	30%
	Memory usage	1000~1050MB
	Running time	42 seconds per epoch
DCGAN	CPU usage	30%
	Memory usage	500MB
	Running time	30 seconds per epoch

Table 4: The benchmarking of DL models on our prototype.

implemented different kernels for SVMs, including linear, linear passive-aggressive, linear ridge, polynomial with 1~10 degrees, and radial basis function (RBF), and principal component analysis (PCA). For CNNs-based attack approaches, we implement based on the framework from VGGnet [36]. For user in-home activities, we implement LSTM-based user in-home activities modeling using Keras model library [17] and TensorFlow framework [3]. Finally, we schedule the batch jobs on our GPU servers to compare the MCC accuracy of 8 different approaches using CUDA. The server that we use to get all the benchmarking and evaluation results for attacking models has resources as follows: 1) CPU: 2x Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, 2) GPU: nVidia TITAN X (Pascal) (x8), 3) RAM: 128GB, 4) OS: Linux CentOS 7. PrivacyGuard can be implemented on IoT hubs and middle-boxes (e.g., Wi-Fi access points, gateway routers, smart IoT hubs).

6 EXPERIMENTAL EVALUATION

Below 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 [33] that includes second 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 process the IoT traffic metadata traces to IoT traffic rate data and also label all the user activities.

Dataset 2: Smart* (omitted for double-blind reviewing). We set up our own "mock" smart home using our lab space that has 4 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 then install *tcpdump* on this gateway to capture network traffic data. We use 45 days 1 minute level IoT traffic data to evaluate our PrivacyGuard.

Dataset 3: Real Smart Homes. We also deploy 22 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, and also record all the groundtruth user activities for two weeks.

Dataset 4: IoT Device Captures (Kaggle #1). We download the IoT Device Captures dataset from Kaggle, and it has 30 IoT devices and each IoT device was recorded for 20 segments and each segment has a traffic duration as of 2 minutes.

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 IoT devices for 5 days using NodeMCU with ESP8266 wifi module.

	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	23.12

Table 5: The correlation and traffic overhead comparison of 3 different major traffic modification approaches.

Dataset 6: User In-home Activity Groundtruth Dataset. 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. We release this groundtruth dataset along with the above-mentioned evaluation datasets.

Network Traffic Rate Data Preprocessing. To learn the effect of traffic rate granularity on user privacy preserving degree, we preprocess the traffic rate traces of the above-mentioned datasets into different granurality levels, such as one second, one minute, three minutes, five minutes, and ten minutes. By default, traffic rate granularity is set as of one second.

6.2 Experimental Setup

Pure Traffic Injection (PTI) Approach. We first implement a general version of prior work [8, 25]. This approach leverages Bernoulli distribution, Poisson distribution, Linear Chain Conditional Random Field (LCCRF) to randomly inject "fake" traffic demands that are randomly selected from historical traffic patterns.

Hybrid Traffic Reshaping (HTR) Approach. We implement a general version of prior work [6, 9]. This approach employs a threshold-based traffic demand flattening, and leverage Bernoulli distribution, Poisson distribution, and Linear Chain Conditional Random Field (LCCRF) to randomly inject "fake" traffic demands.

Random Traffic Padding (RTP) Approach. We implement a general version of prior work [5, 13, 15]. This approach employs traffic demand flattening, and leverage Hidden Markov Model (HMM)-based user behavior modeling to randomly inject "fake" traffic demands that are randomly selected from historical traffic patterns.

PrivacyGuard Approach. PrivacyGuard employs intelligent DCGANs-based IoT device traffic signature learning, Long short-term memory (LSTM)-based artificial traffic signature injection, and partial traffic reshaping to further obfuscate private information that can be externally observed in IoT traffic traces.

PrivacyGuard$^+$ (with optimization) Approach. 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

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

Matthews Correlation Coefficient (MCC). To quantify the accuracy of different user privacy enhancing approaches, we note that the standard evaluating metrics, e.g, accuracy, F1, would not work well on our highly imbalanced IoT traffic data. Based on the recommendation from prior work [4, 27], we use the MCC [21], a

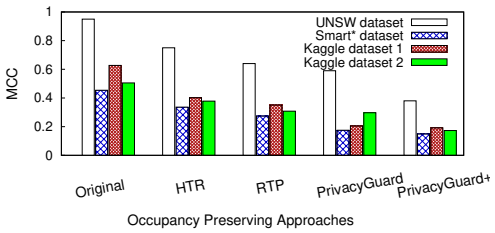


Figure 11: The accuracy comparison of occupancy detection after applying 4 different approaches.

standard measure of a classifier's performance, where values are in the range −1.0 to 1.0, with 1.0 being perfect user activity detection, 0.0 being random user activity prediction, and −1.0 indicating user activity detection is always wrong. The expression for computing MCC is below, 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.

$$\frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{1}$$

Cohen's Kappa. The Cohen's Kappa [10] is a measure of the agreement between two classifiers who each classify N items into C mutually exclusive categories. The Cohen's Kappa is defined as,

$$\kappa = 1 - \frac{1 - p_o}{1 - p_e} \tag{2}$$

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, $\kappa = 0$.

Pearson Correlation Coefficient (PCC). The PCC [26] 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 (SRCC). The SRCC [34] 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 PCC that assesses linear relationships, 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 PCC and SRCC to quantify the effectiveness of different approaches on masking user private information.

Adversary Confidence (AC). We leverage 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 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,

$$AC = \frac{np}{np + n(1 - p)q} \tag{3}$$

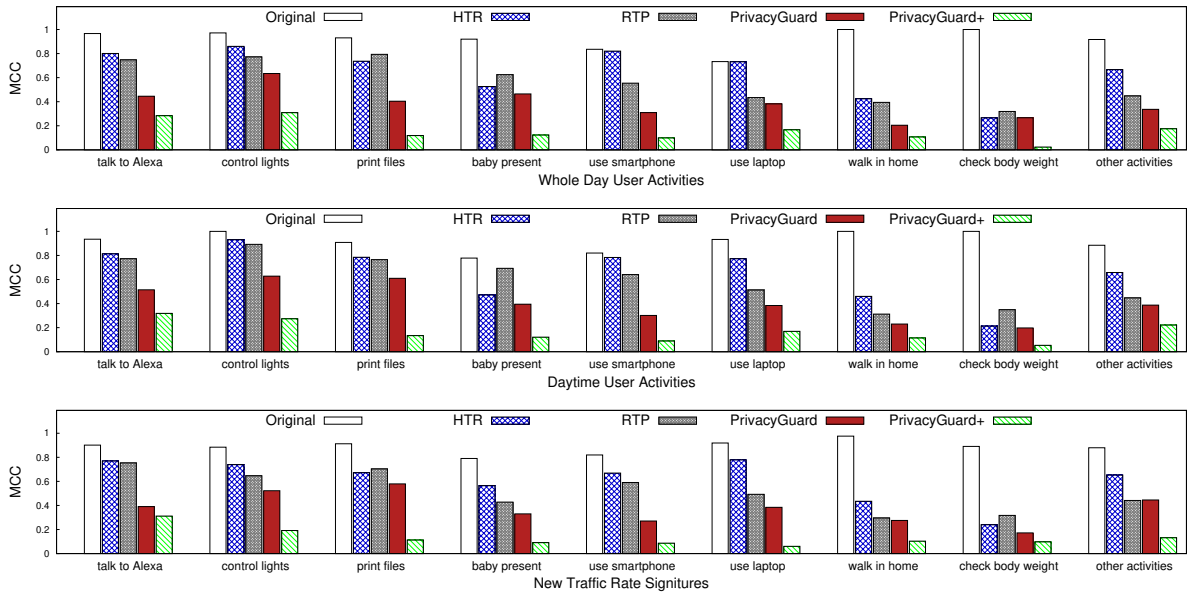


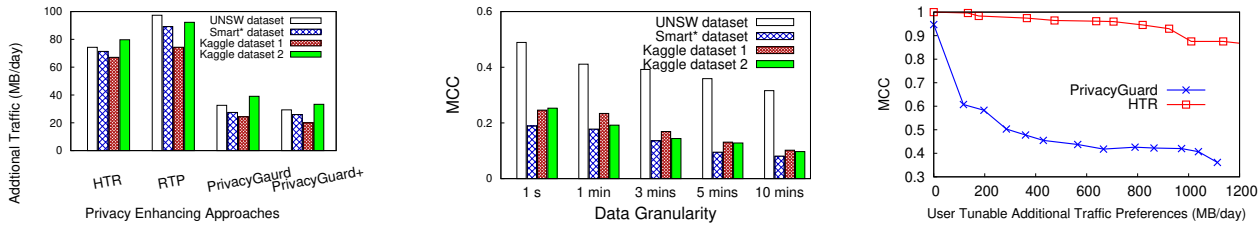
Figure 12: The whole day (top), daytime (middle) and daytime with new traffic patterns (bottom) MCC comparison of user activities detection before (original) and after applying HTR, RTP, PrivacyGuard, and PrivacyGuard$^+$.

6.4 Experimental Results

6.4.1 Preventing Occupancy Detection. We first compare the ability of four different approaches regarding masking occupancy. These approaches split the dataset into training and testing dataset using a ratio of 7:3 after cross-validation. To perform fair comparison, we set traffic "cap" for each approach as 75MB per device per day. As shown in Table 5, PTI receives PCC and SRCC as the value of 0.75 and 0.83, and HTR reports PCC and SRCC as of 0.46 and 0.71, respectively. RTP reports smaller PCC and SRCC as values of 0.58 and 0.69, respectively. While, PrivacyGuard yields the smallest PCC and SRCC as the values of 0.35 and 0.64, respectively. Thus, among all the four different approaches, PrivacyGuard is the best performing approach to hide user occupancy. In addition, 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, PrivacyGuard, and PrivacyGuard$^+$ using MCC. As shown in Figure 11, PrivacyGuard$^+$ yields the average detection MCC as of 0.2235, which is much closer to random prediction, i.e., an MCC of 0.0, and are a factor of >2 times less than the average MCC when attacking on the HTR modified traffic, which are 0.4665.
Results: *By lowering the average MCC to 0.2235 in 4 smart homes, PrivacyGuard$^+$ approach effectively prevents occupancy detection from a wide set of ML-based and DL-based attacks. In addition, PrivacyGuard$^+$ yields a factor of >2 times less than the MCC of occupancy attacks on the modified IoT traffic traces by prior approaches.*

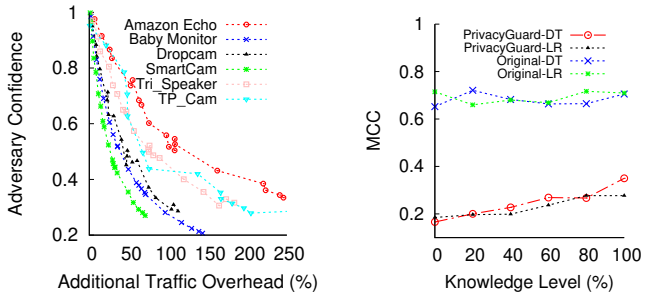
6.4.2 Preventing User Activities Detection Attacks. We next benchmark the effectiveness of masking user activities when applying three different privacy preserving approaches. We leverage ML/DL-based attack models that we built in Section 3 to detect 9 different user activities using the original, HTR modified, PrivacyGuard modified, and PrivacyGuard$^+$ modified traffic rate traces. Unsurprisingly, as shown in Figure 12, 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 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 in the nighttime (12 am to 7 am), and thus most of the traffic occurs in this period are not reflecting user in-home interactive activities. Therefore, all the 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 UNSW dataset and Smart* dataset.

In addition to Figure 12 (top and middle), we also examine the performance of the different reshaping approaches when handling new signatures. The goal is to benchmark these traffic reshaping approaches when the incoming traffic having a mix of known (pre-seeded) and unknown (new) IoT traffic rate signatures. For this example, we set the ratio as 1:1, and we find that all the traffic reshaping approaches achieved the similar MCCs as shown in Figure 12 (middle) which primarily reshaped traffic rate traces using known (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.
Results: *Our PrivacyGuard$^+$ approach effectively prevents 9 different user activities from a wide set of ML-based and DL-based sophisticated attacks in smart homes. Compared with prior approaches, PrivacyGuard$^+$ consistently yields the worst MCC for each user activity, and thus is the best performing privacy preserving approach.*



(a) (b) (c)

Figure 13: Left graph (a) shows the amount comparison of additional traffic when applying different approaches. Middle graph (b) and right graph (c) show the accuracy comparison of user activities detection when applying PrivacyGuard$^+$ on traffic rate data in different granularities and different user "tunable" additional traffic preferences.



(a) Adversary Confidence (b) Adaptive Adversary

Figure 14: PrivacyGuard performance comparison under different adaptive adversaries.

6.4.3 Quantifying Traffic Overhead. We quantify the amount of network traffic overheads that are required to perform HTR, RTP, PrivacyGuard, and PrivacyGuard$^+$. Figure 13 (a) reports the amount of additional traffic consumption for each approach. As expected, PrivacyGuard$^+$ only consumes 27.15 MB traffic per day on average over four datasets, which is ~2.7 times less than that of HTR, which is 73.11 MB on average per day. That says, PrivacyGuard$^+$ consumes the least amount of traffic overhead while achieves the best performance to prevent user privacy leakage in 4 smart homes.
Results: *PrivacyGuard$^+$ only consumes 27.15 MB traffic per day, which is ~2.75 times less than that of HTR, which is 73.11 MB per day. PrivacyGuard$^+$ consumes the least amount of traffic overhead while achieves the best performance to prevent user privacy leakage.*

6.4.4 Quantifying Accuracy When Varying Granularity of Traffic Traces. We next evaluate user activity detection effect on different traffic rate traces that have different level of granularities, 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 traffic traces. As shown in Figure 13 (b), as expected, higher granularity results in lower user activity detecting accuracy in MCC. This is mainly due to the facts that 1) PrivacyGuard$^+$ performs consistently well on different traffic trace data at different granularities, 2) fewer fluctuations and spikes are observed in higher resolution traffic rate traces. In addition, when traffic rate traces are becoming coarser, 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 minutes level network traffic rate traces, which is nearly the same as random prediction, i.e., an MCC of 0.0.*

6.4.5 Quantifying accuracy when varying user tunable reshaping preferences. We then evaluate 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 13 (c), PrivacyGuard significantly reduces user activity detection accuracy—MCC from ~0.95 to ~0.5, after applying users' allowance of additional traffic as 6.45 MB per device per day (equivalent to ~1.83 KB per device per minute). While, 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 2 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 trade-off between user differential privacy preserving and traffic overhead. In addition, when applying additional 11.61 MB per device per day traffic overhead, PrivacyGuard yields an MCC as 0.46, which is 2 times less than the MCC of 0.97 using HTP.*

6.4.6 Preventing User Activities Detection by Adaptive Adversary. We next examine the effect of different adversary confidence (AC) and different level adaptive adversary on PrivacyGuard's performance. As shown in Figure 14 (a), for the top-6 traffic consuming IoT devices, the adversary's confidence drops significantly when PrivacyGuard having more traffic overhead. In particular, the cameras (e.g, baby monitor, drop camera, smart camera) reports the fastest AC decreasing. This is mainly due to the fact that these IoT cameras themselves have more "unstable" patterns than other IoT devices which allow them to better respond to our PrivacyGuard. Figure 14 (b) shows the ability of PrivacyGuard to preserve user privacy when adaptive adversary having more knowledge about our attack model. To generate the average MCC values in Figure 14 (b), we apply the Logistic Regression (LR)-based and Decision Tree (DT)-based attack models on PrivacyGuard modified traffic traces to infer 9 user activities and report the mean value of the MCCs, respectively.

In essence, we define the attacker knowledge-level as the percentages of traffic rate testing dataset that an external adversary can leverage to train or calibrate its attack models to infer user in-home

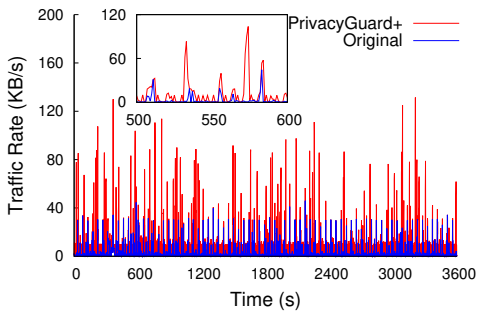


Figure 15: Masking user activities with PrivacyGuard prototype. The zoomed-in inset shows PrivacyGuard's online operations to hide user in-home traffic patterns.

activity. 0% indicates that the external adversary has no knowledge of the target home testing dataset and no cross-validation is performed to train the attack models. While, 100% means that the external adversary has observed all the groundtruth traffic patterns for each user activity so that the attack models are "perfectly" trained. The goal of this experiment is to understand PivacyGuard's ability to protect user private information from the attack initialized by "adaptive" adversaries who have different groundtruth knowledge levels from the target smart home. We find that PrivacyGuard modified traffic's MCC slightly increases from 0.16 to 0.35, while original traffic's MCC is fluctuating between 0.65 and 0.72. This is because attacking original traffic to infer user in-home activity is surprisedly easy as we showed in Section 2 and Section 3. Note that, even when an adversary having 100% knowledge about PrivacyGuard's deep learning model, PrivacyGuard still can prevent data analytics attacks as the MCC of 0.35, which is the almost 2 times less than the original traffic's MCC as of 0.72.

Results: *Using PrivacyGuard, the adversary's attack confidence significantly drops when user permitting additional overhead. Also, PrivacyGuard yields an MCC of 0.35 which is almost 2 times less than original traffic's MCC when an adversary having 100% knowledge of our PrivacyGuard's modeling.*

6.4.7 Prototype Demonstration. Figure 15 demonstrates the performance of PrivacyGuard$^+$ prototype for an 3,600 seconds (1 hour) period online traffic rate data (in KB/s). 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 to a 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 traffic reshaping, to mask the traffic usage trends exposed in traffic rate traces. Ultimately, our prototype demonstrates that PrivacyGuard$^+$'s approach permits a straightforward implementation using widely-used, off-the-shelf components. Also, as shown in Table 6, PrivacyGuard can enable users to significantly reduce privacy leakage, while still permitting regular IoT device usage.

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

Table 6: IoT device response time with and without PrivacyGuard.

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.*

7 CONCLUSION AND FUTURE WORK

We design 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 evaluate PrivacyGuard using IoT network traffic traces of 31 IoT devices from 5 smart homes and deploying a Raspberry Pi 4-based prototype. We find that PrivacyGuard can effectively prevent a wide range of state-of-the-art machine learning-based and deep learning-based occupancy and other 9 user in-home activity detection attacks.

We plan to collect more IoT traffic traces to further understand the effect on the trade-off between privacy preserving and traffic overhead. In addition, we also plan to develop a tailored smart router operating system that can host PrivacyGuard services directly.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and our shepherd for providing us their insightful comments and valuable feedback, which significantly improved the quality of this paper. This research is supported by Cyber Florida Collaborative Seed Program.

REFERENCES

[1] 2020. Ostinato: Packet Generator and Network Traffic Generator. https://ostinato.org/.
[2] 2021. PrivacyGuard. https://github.com/cyber-physical-systems/PrivacyGuard.
[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 265–283.
[4] Josephine Akosa. [n.d.]. Predictive accuracy: a misleading performance measure for highly imbalanced data.
[5] 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.
[6] Phuthipong Bovornkeeratiroj, Srinivasan Iyengar, Stephen Lee, David Irwin, and Prashant Shenoy. 2020. RepEL: A Utility-preserving Privacy System for IoT-based Energy Meters. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 79–91.
[7] T. Brewster. 2017. Now Those, Privacy Rules Are Gone, This Is How ISPs Will Actually Sell Your Personal Data. https://www.forbes.com/sites/thomasbrewster/2017/03/30/fcc-privacy-rules-how-isps-will-actually-sell-your-data/.
[8] 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, 227–238.
[9] Dong Chen, David Irwin, Prashant Shenoy, and Jeannie Albrecht. 2014. Combined heat and privacy: Preventing occupancy detection from smart meters. In *2014 IEEE International Conference on Pervasive Computing and Communications*. 208–215.
[10] CKA [n.d.]. Cohen's Kappa. https://en.wikipedia.org/wiki/Cohen%27s_kappa.
[11] 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, 43–48.

[12] Wenbo Ding and Hongxin Hu. 2018. On the Safety of IoT Device Physical Interaction Control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. 832–846.
[13] 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 *2012 IEEE symposium on security and privacy*. IEEE, 332–346.
[14] Md Kamrul Hasan, Husne Ara Rubaiyeat, Yong-Koo Lee, and Sungyoung Lee. 2008. A reconfigurable HMM for activity recognition. In *2008 10th International Conference on Advanced Communication Technology*, Vol. 1. IEEE, 843–846.
[15] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security*. Springer, 27–46.
[16] 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 *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 232–240.
[17] Nikhil Ketkar. 2017. Introduction to keras. In *Deep learning with Python*. Springer, 97–111.
[18] Jinyang Li, Zhenyu Li, Gareth Tyson, X Gaogang, et al. 2020. Your Privilege Gives Your Privacy Away: An Analysis of a Home Security Camera Service. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE.
[19] Nicole Lindsey. 2019. Smart Devices Leaking Data to Tech Giants Raises New IoT Privacy Issues. https://www.cpomagazine.com/data-privacy/smart-devices-leaking-data-to-tech-giants-raises-new-iot-privacy-issues/.
[20] 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.
[21] mcc [n.d.]. Matthews Correlation Coefficient. https://en.wikipedia.org/wiki/Matthews%_correlation%_coefficient.
[22] Mirimir. 2018. Collection of User Data by ISPs and Telecom Providers, and Sharing with Third Parties. https://www.ivpn.net/blog/collection-of-user-data-by-isps-and-telecom-providers-and-sharing-with-third-parties.
[23] mozilla 2019. ISPs Lied to Congress to Spread Confusion about Encrypted DNS, Mozilla says. https://arstechnica.com/tech-policy/2019/11/isps-lied-to-congress-to-spread-confusion-about-encrypted-dns-mozilla-says/.
[24] 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, 131–134.
[25] 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.
[26] pcc [n.d.]. Pearson Correlation Coefficient. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.
[27] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. 2018. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. (2018).
[28] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).
[29] Vasanthan Raghavan, Greg Ver Steeg, Aram Galstyan, and Alexander G Tartakovsky. 2013. Coupled hidden markov models for user activity in social networks. In *2013 IEEE International Conference on Multimedia and Expo Workshops*.
[30] 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.
[31] Karsten Rothmeier, Nicolas Pflanzl, Joschka Hüllmann, and Mike Preuss. 2020. Prediction of Player Churn and Disengagement Based on User Activity Data of a Freemium Online Strategy Game. *IEEE Transactions on Games* (2020).
[32] Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*. Springer, 18–33.
[33] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing* (2018).
[34] srcc [n.d.]. Spearman's Rank Correlation Coefficient. https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient.
[35] Statista. 2016. Internet of Things Connected Devices Installed base Worldwide from 2015 to 2025 (in billions). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.
[36] vgg [n.d.]. Very Deep Convolutional Networks for Large-Scale Visual Recognition. https://www.robots.ox.ac.uk/~vgg/research/very_deep/.
[37] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*. 143–157.
[38] Tao Wang and Ian Goldberg. 2016. On realistically attacking tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies* 4 (2016), 21–36.
[39] Tao Wang and Ian Goldberg. 2017. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium*. 1375–1390.
[40] 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.
[41] 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* (Alexandria, Virginia, USA) *(CCS '08)*. 323–332.