PAROS: The Missing "Puzzle" in Smart Home Router Operating Systems

Keyang Yu
Department of Computer Science
Colorado School of Mines

Dong Chen
Department of Computer Science
Colorado School of Mines

***Abstract*—The Internet of Things (IoT) devices have been increasingly deployed in smart homes for automation. Unfortunately, extensive recent research shows that external on-path adversaries can infer and fingerprint user sensitive in-home activities by analyzing IoT network traffic rates alone. Most recent traffic padding-based defending approaches cannot sufficiently protect user privacy with reasonable traffic overhead. In addition, these approaches typically assume the installation of additional hub hardware in smart homes to host their traffic padding-based defending approaches. To address these problems, we design a new open-source traffic reshaping system—privacy as a router operating system service (PAROS) that enables smart home users to significantly reduce private information leaked through IoT network traffic rates. PAROS does not assume the installation of any additional hardware device. We evaluate PAROS on open-source router Operating System (OS)—OpenWrt enabled virtual machine and also two real best-selling home routers. We find that PAROS can effectively prevent a wide range of state-of-the-art adversarial machine learning-based user in-home activity inference attacks, with near-zero system overhead increasing.**

***Index Terms*—Internet of Things, User Privacy, Machine Learning, Router OS, Data Analytics**

I. Introduction

People are increasingly deploying the Internet of Things (IoT) devices for their smart home automation. The total installed base of IoT devices is projected to 30.9 billions worldwide by 2025, a sharp jump from 13.8 billion units in 2021 [32]. Traffic data generated by 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. Verizon uses "supercookies" to track 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 [20]. ISPs like AT&T, Comcast, Sprint, and Verizon are selling personal traffic data without prior user consent to "enhance" user experience [8]. Moreover, recent IoT privacy survey [17] shows that 72 out of 81 popular IoT devices are sharing data with third-parities (e.g., Google, Amazon, Akamai) completely unrelated to original manufacturer and far beyond basic necessary device configuration, including voice speakers, smart TVs, and streaming dongles. Meanwhile, significant recent research [6], [7], [9], [10], [12], [13], [15], [23], [30], [36], [37] shows that launching user activities inference attacks is surprisingly easy, since user activities highly correlate with simple time-series data statistical metrics. Thus, IoT device traffic rates alone have significant user privacy threats. To address this privacy issue, significant recent work [6], [9], [11], [21], [34]–[36], [38] proposes traffic reshaping-based prevention techniques to thwart privacy attacks on IoT traffic rates. Unfortunately, these approaches can not sufficiently protect user privacy with reasonable traffic overhead. Specially, these approaches typically assume the pre-installation of an additional hub hardware (e.g., Raspberry Pi, IoT Hub) [2], [38], which is not always available in every home, to host their traffic padding approaches.

To address these problems, we design a new open-source traffic reshaping system—privacy as a router operating system service (PAROS) that enables smart home users to significantly reduce the private information leaked through IoT device network traffic rates. PAROS does not assume the pre-installation of any additional hardware in a smart home. In doing so, this paper makes the following contributions.

Challenges. We explore and highlight the major challenges to design efficient user privacy preserving approaches on the resources limited home routers directly.

PAROS Design. We present the design of PAROS, which enables users to re-gain the control on reducing their privacy leakage through IoT network traffic. In essence, PAROS leverages traffic rate signature learning, hidden Markov model (HMM)-based artificial traffic signature injection, and partial traffic padding to obfuscate user privacy. We also design a Support Vector Machines (SVM)-based memory replacement scheme to further optimize PAROS's performance.

Implementation and Evaluation. We implement PAROS both simulator and new kernel service using the most widely used programming language—`C` for router operating systems and firmwares. We evaluate and benchmark PAROS on open-source router OS—OpenWrt enabled virtual machine and also two real commercial router deployments. We find that PAROS can effectively prevent a wide range of state-of-the-art adversarial machine learning-based user activities inference attacks, with near-zero system overhead increasing.

Releasing Datasets and Code. Our approaches to prevent user sensitive information leakage through IoT traffic rates are quite general, and can be applied to address similar user privacy problems. We release the source code and datasets of PAROS to IoT research community on our website [4].

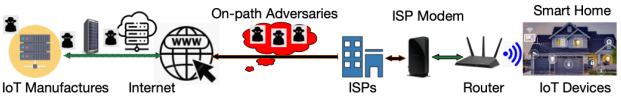

Fig. 1. Overview of our privacy threat model in a smart home.

II. Background and Related Work

A. Privacy Threat Model

As shown in Figure 1, we are broadly concerned with the ability of external adversaries (e.g., ISPs, on-path network observers, manufactures, and their 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 and package sizes, 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 external adversaries can use sophisticated user information inference techniques, such as machine learning (ML), deep learning (DL), or other statistical methods to infer certain types of the observed user pattern information in the recorded traffic rates. Thus, inferring or discovering user activities in these homes is considered as an opposition to users' privacy preferences.

In particular, we are concerned with four types of user privacy inference attacks: i) *Learning user occupancy from the data.* This includes whether a home or building is occupied and when; ii) *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; iii) *Learning user in-home activities from the traffic data.* These user activities are inferred using IoT device activities and 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, playing online games; iv) *Learning user contextual information from the data.* In addition to the above-mentioned short-term user activities, adversaries may also infer more comprehensive and longer-term user activities, such as whether a household has a baby, and whether they go on vacation on weekends.

Attack Scenario #1: To infer the type of IoT devices 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 to learn whether a particular IoT device (e.g., Voice Assistant) is present in a home and what model of an IoT device is present. Then, the external attacker may launch cyberattacks for a specific IoT device when user daily routine permits.

Attack Scenario #2: An external adversary from ISPs, IoT device manufacturers or their third-parties is actively monitoring the IoT traffic traces to learn user activities and uses data analytic approaches to learn indirect user privacy information (e.g., user short-term and long-term activities) that might be interesting for insurance companies, marketers, or government.

We also assume our smart home users trust in Amazon AWS (EC2) or Google Cloud services to host their remote servers to protect their 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.

B. Related Work

There is a gap in the literature concerning privacy preserving methodologies that use ML/DL techniques and are deployed in home routers. We outlined the design alternatives that are most related to our work. In doing so, we review a wide range of the most recent sophisticated traffic reshaping-based prevention techniques [6], [7], [9], [10], [13], [14], [23], [27], [28], [30], [36], [37] to thwart privacy attacks on IoT traffic rates. To understand the performance of different approaches, we implemented three different traffic reshaping approaches. Table I quantifies the effectiveness of the seven recent approaches. We use ϵ-security [23] to describe the probability of a traffic reshaping approach can not prevent users from external adversarial inferring attacks.

Pure Traffic Injection. Prior work [9], [18], [23] proposed defense approaches to inject artificial network traffic patterns to conceal genuine user network traffic patterns. Table I shows the general implementation of these approaches yield additional overhead as $\sim$97%. Park et al. found that traffic traffic encryption cannot prevent privacy invasions exploiting traffic pattern analysis and statistical inference [23]. Park et al. developed empirical models to statistically learn user behaviors from wireless sensors status transition data. Then, cloaking network traffic patterns are injected to obscure genuine traffic patterns. Cai et al. presented a defense—Tamaraw against Tor website fingerprinting that can reshape traffic rate traces by controlling the size of the parameter to pad packets [9]. However, these approaches did not always hide the genuine network traffic patterns, in particular, during higher and lower traffic rate periods. In addition, their traffic injection process was built in a simulator which would require a device to host it to reshape traffic rates. The computation, communication, and storage costs are not fully evaluated towards real deployment.

Random Traffic Padding. Recent work [6], [13], [15] proposed random traffic padding-based defending approaches that could prevent an external adversary from reliably distinguishing genuine user involved traffic patterns from "fake" traffic patterns. As shown in Table I, the general implementation yields additional overhead as $\sim$165.9%. Dyer et al. proposed a buffered fixed-length obfuscator based 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]. They matched the gaps between traffic packets with a distribution of generic network traffic. Similarly, Apthorpe et al. presented a stochastic traffic padding algorithm, which can be deployed on edge gateways, middle boxes, or IoT hubs to flatten traffic patterns and inject fake traffic patterns that look like the real IoT traffic patterns [6]. Due to computing and storage

	Additional Hardware	Security (ϵ)	Additional Overhead
Pure Injection	Yes	87.15%	97%
Random Padding	Yes	54.33%	165.9%
Hybrid Reshaping	Yes	72.6%	103.7%
Tor [9]	Yes	77.5%	25%
RepEL [7]	Yes	33%	100%
Tamarow [9]	Yes	3.4%	199%
PrivacyGuard [38]	Yes	14.2%	66.7%

TABLE I
THE COMPARISON OF SEVEN MAJOR TRAFFIC RESHAPING APPROACHES.

cost, these approaches typically require the deployment of additional hardware to perform their traffic padding operations.

Hybrid Traffic Reshaping. Prior work [7], [10], [14], [26]–[28], [30], [37], [38] presented hybrid reshaping techniques to prevent user privacy leakage . These approaches typically combined traffic rate flattening and artificial traffic signature injection to further obscure user privacy. Table I shows the general implementation yields additional overhead as $\sim$103.7%. Chen et al. proposed to learn the "noise" injection rate using empirical analytics of IoT device activities [10]. Similarly, Bovornkeeratiroj et al. proposed RepEL which employed an edge gateway (typically, a Raspberry PI-class node) to partially flatten traffic loads and randomly replay traffic loads to hide user occupancy information [7]. Shmatikov et al. proposed adaptive padding algorithms to destroying timing "fingerprints" application traffic by enforcing inter-package intervals to match pre-defined probability mass functions [30]. Wang at al. [37] designed a traffic padding algorithm that uses matched package schedules to prevent adversaries from paring incoming and outgoing traffic. Significant work [14], [26]–[28] proposed to model user activities using Markov Chain model. Keyang et al. proposed PrivacyGuard [38] assumed the installation of an additional IoT hub to reshape both incoming and outgoing traffic concurrently. Due to the nature of empirical modeling and random injection, these approaches may still allow external attackers to identify the injected "fake" signatures, and thus infer genuine user activities. These approaches typically assumed the installation of either a simulator (on a computer) or edge gateway (typically, a Raspberry PI-class node) to enable their defending approaches.

Observation. Table I shows that random traffic padding approach—Tamarow yields the lowest ϵ-security as of 3.4%. Unsurprisingly, pure traffic injection approach reports the highest ϵ-security as of 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 any real IoT traffic patterns already presented in IoT traffic traces. Hybrid traffic reshaping approach reports better ϵ-security (e.g., PrivacyGuard [38], RepEL [7]). Hybrid traffic reshaping approach also makes its best efforts to partially flatten both genuine and "fake" traffic patterns. The different correlation performance between hybrid traffic reshaping approach and random traffic padding approach is that random traffic padding approach generally has higher flattening threshold to pad IoT traffic patterns, and also consider bidirectional traffic padding for IoT devices (e.g., Amazon Alexa, Google Home). For the same reason, Tamarow [9] reports the maximum traffic overhead as of 199% additional overhead per device per day. Although prior work has improved user privacy preserving, many of these work assumed the installation of an addition hardware to "host" their reshaping approaches.

C. Summary

Prior research proposes significant work to thwart privacy attacks on IoT traffic rates. Unfortunately, these approaches still have the following limitations towards real home deployments. (1) Many existing approaches are proposed, implemented and evaluated on a simulator which is potentially installed on a host device (e.g., desktop, middle box, edge gateway, WiFi access point). Thus, these approaches are not (fully) evaluated in a real smart home yet. (2) The most recent notable approaches (e.g., [6], [38]) assumed the installation of an additional hardware—IoT Hub, which is running on a open-source firmware or having adaptive application programming interfaces (APIs), to enable its traffic shaping algorithms. (3) Many recent work leveraged standard ML/DL frameworks to build their defending approaches. Their model (re)training process is typically heavy for standard smart home IoT hubs, edge gateways or middle boxes. Also, full stack evaluation (e.g., energy consumption, memory storage, CPU utilization) towards read-world deployment is missing. Thus, new practical approaches are necessary. These valuable insights will guide the development of our proposed technique—PAROS.

III. CHALLENGES

In this section, we outlined the major challenges we met when designing and evaluating our PAROS.

No Additional Device. Many prior defending approaches [13], [15], [38] required the deployment of additional hardware (e.g., Amazon Alexa, Google Home, Raspberry Pi), which is running on open-source firmware or adaptive application programming interfaces (APIs), to reshape traffic rates of a smart home. And they also assume they could apply any changes that are necessary to build their new approaches. Unfortunately, this is not the case in real practice. To address this issue, PAROS only leverage existing popular IoT or edge devices in smart homes and does not assume any additional hardware installation. Specially, PAROS identifies the most common devices—home routers to host new defending approaches.

Unexpected Service Latency. The defending techniques hosted on IoT hubs may cause IoT service delay. The most recent work [38] has shown that Amazon Alexa, Google Home, and Belkin Switch have 0.51, 0.47, and 0.76 seconds latency on their cloud services after applying the most notable traffic reshaping approach [38] on their IoT hub, respectively. This service latency may disimprove smart home user experience. To address this issue, PAROS is designed in a lightweight manner and also leverages multiple optimization techniques to further reduce traffic reshaping overhead. In doing so, PAROS can significantly obfuscate private information that can be observed in IoT traffic trace with reasonable traffic overhead.

Limited Computing Resources. The most common devices in smart homes are home routers. Different from IoT hubs or edge gateways, the best selling home routers typically have very limited computing resources. For instance, the Amazon best selling home router—Netgear R6700 has a Broadcom CPU of 1 GHz, 256 MB Random Access Memory (RAM), and 128 MB ROM. These home routers may not be able to (re)train and test standard/heavy ML/DL models directly. In addition, these home routers typically do not have enough memory space to store and manage all the IoT device traffic rate signatures which are the foundations for existing privacy leakage preventing approaches [6], [7], [9], [10], [12], [13], [15], [23], [30], [36], [37]. To address this issue, PAROS leverages a tailored lightweight ML approach and a new memory space replacement approach to learn and replay IoT traffic patterns to build privacy-preserving approaches.

Isolated Ecosystem. Although many IoT devices have their programming APIs, such as Amazon Alexa and Google home, their ecosystems tend to be isolated. In addition, we cannot suspend back-end data collection or sharing. Uploading traffic rates to remote servers may cause privacy preserving concerns. To address this issue, PAROS goes beyond the IoT hubs and work on home routers to prevent user privacy leakage.

No Advanced Development Support. Rather standard system development support, we are not aware of ready-to-use open-source frameworks to support router level ML/DL system development. The most common firmware for home routers is OpenWrt, which is an open-source operating system for embedded operating systems based on Linux and primarily used on embedded devices to route network traffic. To address this issue, we re-design and benchmark multiple alternative ML/DL models to enable intelligent traffic reshaping approaches on router OS using `C` language.

IV. Design

A. System Design

We design a new system—privacy as a router operating system service (PAROS) on OpenWrt OS [22] enabled home routers. Figure 2 shows the system model to build PAROS. OpenWrt is a highly extensible GNU/Linux distribution for embedded devices, typically wireless routers. Unlike other distributions, OpenWrt is built from the ground up to be a full-featured, easily modifiable operating system (OS) for routers. It provides a fully writable file system along with package management, which allows high level user customization and wildly applicable for different system architectures. OpenWrt provides reliable abstraction to the hardware, which make it suitable to deploy similar kernel modules on both X86 and ARM processor architectures [22]. Comparing with other available home router OSes (e.g., DD-WRT), OpenWrt has better version control and higher flexibility on customization. Note that, PAROS can also be deployed on a home gateway or middle box. Home router (with PAROS) is then connected to Internet through Internet Service Providers (ISPs). Figure 3 shows the operational pipeline of PAROS. In essence, PAROS first automatically segments and learns IoT device traffic rate signatures. Then, PAROS leverages a new lightweight hidden Markov model (HMM) to model user in-home behaviors, which will guide the next artificial traffic signature injection process. PAROS also employs a partial traffic reshaping approach to further obscure user private information exposed in IoT traffic rates. Eventually, PAROS provides a full stack performance evaluation and benchmarking for real home router deployments.

Fig. 2. The system model of our PAROS.

B. Intelligent IoT Traffic Signature Learning

PAROS first learns network traffic rate signatures from a smart home. Since PAROS is built in home router OS—OpenWrt, it is very trivial for PAROS to detect smart home IoT devices. Note that, although PAROS can report manufacture brand for each IoT device using MAC address lookup API [3], PAROS only extracts minimum necessary information from each IoT device, which is only segmenting and assigning each IoT device a system identifier (ID) so that PAROS can reshape that device's traffic rates. Different traffic patterns from the same device are distinguished by pattern ID. The traffic pattern information contains `IP.internal` and `IP.external`, the internal IP normally remains same unless the user reboots the router. The external IP indicates the address that the IoT device contacting to, normally some address from the device manufacturer's server. The Duration column indicates the length in second of this particular traffic pattern. It may not be bounded by the TCP handshake since the device may have several ports generating traffic simultaneously, which contains more than one TCP/IP connections. The direction marks IoT traffic as incoming or outgoing traffic, which 0 indicates IoT device is sending traffic to remote server, and 1 indicates for receiving traffic from it. The goal of this traffic rate signature learning process is to ensure that external adversaries cannot reliably distinguish the genuine IoT traffic rate signatures from the later "artificially" replayed traffic rate signatures.

We store all the traffic signatures for IoT devices in a SQLite database using `C` Interface. PAROS also takes additional steps to ensure it is reliably difficult for external adversaries to distinguish artificial traffic rates from real traffic rates. For instance, PAROS also statistically learns the time, duration, and also the fraction of traffic signatures in each category rate levels (e.g., short, long, high, low, medium). PAROS uses this fraction to weight each category's future traffic rate signature selection, such that the "artificial" traffic demand matches the breakdown of real traffic demand. To eliminate the duplicated traffic rate signature learning, PAROS leverages both Dynamic Time Warping (DTW) approach [29] and Pearson Correlation Coefficient (PCC) [24] to quantify similarity between existing times series traffic rate signatures and "new" traffic signatures.

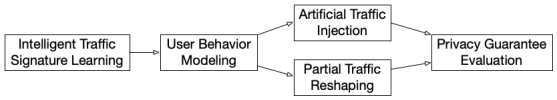

Fig. 3. The system pipeline of our PAROS.

Note that, PAROS examines incoming traffic rate signatures in the same manner, despite whether they are "old" or "new" ones. Thus, PAROS can automatically learn the new traffic rate "patterns" generated by the already connected IoT devices, and also the traffic rate signatures from newly online IoT devices which PAROS has never seen before.

C. User In-home Behavior Modeling

PAROS does not randomly inject artificial traffic rate signatures, since external adversaries may still distinguish "fake" traffic rates from the genuine ones. For instance, it is obvious for external attackers to filter out "fake" randomly injected traffic rate patterns of Amazon Alexa and Google Nest home from 2 am to 6 am. Another example, people tend to use washer before dryer, and thus earlier injected dryer event might be "fake". PAROS carefully replays traffic rate signatures learned in Section IV-B when user in-home behaviors permit.

Prior approaches [6], [38] have explored the benefits of user behavior model guided privacy preserving approaches using Bernoulli distribution, Poisson distribution, Linear Chain Conditional Random Field (LCCRF) and Long short-term memory (LSTM) into their traffic "noise" injections into IoT traffic traces. However, as we discussed in Section III, home routers typically have very limited hardware resources and thus might not be the good candidates to run these heavy ML/DL models. To further evaluate this situation, we benchmark the top three models, including hidden Markov model (HMM), Long short-term memory (LSTM), and convolutional neural network (CNN), which have been widely used in prior work. Figure 4 shows the training cost comparison results of HMM, LSTM and CNN model on UNSW dataset, which has 432,000 seconds daily usage traffic rates for 5 days in a smart home. We find that HMM yields the shortest training time while can still achieve a similar accuracy as other models'.

Our insight is that user in-home behavior can be derived or modeled from the occurrence sequence of multiple IoT device events in a smart home. For instance, a smart home IoT hub (i.e., Amazon Echo, Google Nest Home) can be configured to control multiple plugs and switches, which creates a combination of IoT device events. Motion sensors on smart doors and windows can also trigger the recording of security camera video recording, which can be identified as another combination of simultaneous IoT device events. In addition, different IoT device events may be triggered by the same user in-home activities but at different sequence and frequency. And the whole house traffic rate is the aggregation of these IoT event combinations and thus can be used to learn user in-home behavior. Although Markov Model (MM) reveals direct connections of IoT device status, it requests user have the expertise to explicitly define all the event combinations. In contrast, Hidden Markov Model (HMM)

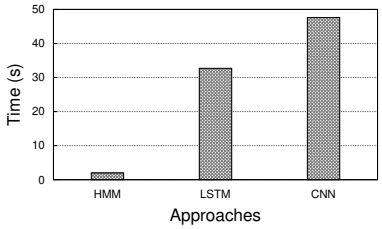

Fig. 4. The cost comparison when training HMM, LSTM, and CNN.

allows user to define a set of hidden status, which might not have obvious connection with IoT device observations. Thus, PAROS leverages HMM model to solve the user in-home behavior learning problem. The learned HMM model will guide later network traffic rate injection process.

However, HMMs, which are based on brutal-force approach and forward-backward algorithm typically yield the time complexity as of $O(Tn^T)$, where n is the number of hidden user behaviors in a smart home, and T indicates the length of the observed IoT device events. To address this issue and also optimize the performance of HMM approach, we re-design the HMM using `C` language which can actually achieve $O(n^2T)$ using Baum-Welch algorithm. Baum-Welch algorithm is used to find the maximum likelihood estimate of the unknown parameters for HMM modeling. Comparing with heavy LSTM or CNN modeling, Baum-Welch inspired HMM approach does not require additional libraries and iterations. The detailed algorithm of HMM approach is established in Algorithm 1.

D. Artificial Traffic Rate Signature Injection

Next, we explain how PAROS leverages HMM-based user behavior model to inject artificial traffic rates into genuine traffic rates. We first classify IoT devices into two categories, including unidirectional and bidirectional traffic IoT devices.

Unidirectional Traffic Rate Injection. It is quite straightforward for PAROS to mimic and inject outgoing traffic for unidirectional traffic IoT devices. These devices may include temperature sensors, humidity sensors, personal weather stations, sleep sensors, security cameras, etc. Most of these IoT devices transmit traffic rates to their IoT hubs or remote servers to provide user service with the collected data.

Bidirectional Traffic Rate Injection. People are increasingly deploying bidirectional traffic IoT devices in their homes. These IoT devices may include voice speaks (e.g., Amazon Alexa, Google Nest Home), Schlage door locks, Drop camera, Google Nest, Honeywell thermostat, etc. However, only very few prior approaches considered to reshape bidirectional traffic rates. The most recent approach—PrivacyGuard [38] used a third-party traffic and package editor and generator—Ostinato [1], which supports multiple communication protocols to generate both outgoing and incoming traffic for bidirectional traffic IoT devices. PrivacyGuard was deployed both on Raspberry Pi and a remote server (with the mixture of Master-Slave and Publish-Subscribe model). Unfortunately, Ostinato is not available on resource limited home routers.

To address this issue, we develop a traffic generation tool that can ping public domains, send request to public

Algorithm 1: The User Behavior Modeling Algorithm

Initialization(Θ_0, $\{O_{1:T}\}$)

Looping()

for $l = 1, \ldots, l_{\max}$ **do**

1. Forward-Backward calculations:

$$\alpha_1(i) = \pi_i b_i(O_1),\ \beta_T(i) = 1,$$
$$\alpha_t(i) = \Big[\sum_{j=1}^{K} \alpha_{t-1}(j) a_{ji}\Big] b_j(O_t),$$
$$\beta_t(i) = \sum_{j=1}^{K} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$
$$\text{for } 1 \le i \le K,\ 1 \le t \le T-1$$

2. E-step:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{K} \alpha_t(j)\beta_t(j)},$$
$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}\beta_t(j))}$$
$$\text{for } 1 \le i \le K,\ 1 \le j \le K,\ 1 \le t \le T-1$$

3. M-step:

$$\pi_i = \frac{\gamma_1(i)}{\sum_{j=1}^{K} \gamma_1(j)},\ a_{ij} = \frac{\sum_{t=1}^{T} \varepsilon_t(i,j)}{\sum_{k=1}^{K}\sum_{t=1}^{T} \varepsilon_t(i,k)},$$
$$w_{kd} = \frac{\sum_{t=1}^{T} \gamma_t(k,d)}{\sum_{t=1}^{T}\sum_{r=1}^{D} \gamma_t(k,r)}$$
$$\text{for } 1 \le i \le K,\ 1 \le j \le K,\ 1 \le k \le K, 1 \le d \le D$$

end

Result($\{\Theta_l\}_{l=0}^{l_{\max}}$)

service APIs (e.g., National Weather Service (NWS) API, OpenWeather API) and upload data to remote servers from home routers using C language to mimic outgoing traffic flows. Similar to PrivacyGuard [38], we also deploy a remote server to mimic incoming traffic flows for bidirectional traffic IoT devices. The remote server can be shared by multiple IoT devices and run traffic editors to change the source IP/MAC address in the pretend incoming traffics at run-time. In doing so, PAROS is able to mimic incoming traffic from the source "valid" IoT remote servers. In addition, PAROS selects traffic rate signatures from SQLite database to inject at an injection rate equal to the rate at which the home generates traffic when the house is occupied. The final goal of this traffic injection is to ensure the injected traffic patterns still fit the traffic distributions that represent the regular user in-home behaviors so that external adversaries cannot reliably distinguish the injected traffic patterns from the genuine traffic patterns.

E. Partial Traffic Rate Padding

The prior step—HMM guided artificial traffic rate injection has obscured most of the private information that are exposed in the external observed traffic rates. However, as discussed in Section II), the modified traffic rates might still expose some minor spike changes that can be potentially identified by external adversaries. Significant prior work [7], [9], [10], [12], [13], [15], [23], [30], [36], [37] aimed at injecting more traffic rate so that external observed traffic rates were "flat" and thus hided more user private information. Unfortunately, due to network traffic generator/editor latency, it is not feasible to perfectly flatten traffic rates. To address this issue, we proposed an additional partial traffic padding approach. Instead of flattening traffic rates, we will reshape the traffic rates using dynamic padding thresholds and extend reshaping periods for random seconds. Note that, PAROS can automatically learn an default trade-off point where smart home users can achieve the "best" protection of their in-home privacy (in terms of ϵ-security [23]) using the "least" additional traffic overhead. In addition, for people who would prefer to trade more additional traffic overhead for a better privacy guarantee, PAROS can automatically finetune its partial traffic padding thresholds to hide more private information in their home traffic rates.

F. Full Stack Privacy Guarantee Evaluation

PAROS provides a full stack privacy guarantee related evaluations. In essence, PAROS first could evaluate the performance of privacy guarantee using multiple metrics, including Matthews Correlation Coefficient (MCC) [19], Pearson Correlation Coefficient [24], and Adversary Confidence (AC) [38]. Secondly, PAROS also supports attack evaluations from adaptive adversaries who may gain advanced knowledge levels about a smart home after monitoring it for a long term. Eventually, PAROS also use Wemo smart plug sensors to support energy-aware privacy guarantee evaluation that beyonds prior smart home privacy defending approaches' evaluations. Note that, although PAROS currently leverages lightweight HMM approach to model users in-home behaviors, we can also potentially integrate with other lightweight ML/DL models to achieve a more comprehensive understanding of user behaviors in a smart home. However, evaluating the performance of these "tiny" models is out of the scope of this work now and will be investigated in our future work.

V. Optimization

As we discussed in Section III, home router typically have very limited memory hierarchy systems. We outlined the optimization techniques we design to mitigate this issue.

A. Limited Memory for Traffic Rate Signatures

To understand this limitation, we first benchmark a smart home that has 30 IoT devices for 5 days. We find that RAM capacity almost has a linear relationship (45.35 KB per device) with the number of IoT devices. In nowadays, people can get $\sim$139 IoT devices from Amazon, which can result in higher demand on RAM capacity. To mitigate this problem, we propose a new memory data replacement approach.

B. SVM Assisted Memory Data Replacement

To minimize the RAM load on home routers, PAROS leverages a new memory data replacement approach. We first implement and benchmark the alternative approaches—First

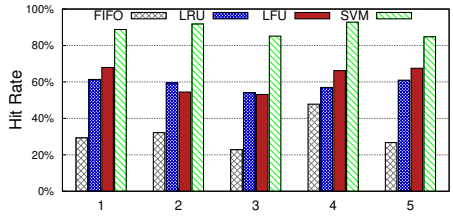

Fig. 5. The comparison of FIFO, LRU, LFU, and SVM assisted memory data replacement approaches on 5 different experiments.

Come First Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), and SVM using C language. Figure 5 shows the comparison results when applying FIFO, LRU, LFU, and SVM on router memory data replacement process. We find that SVM yield the least amount of RAM space. Thus, we select SVM to assist PAROS' memory management.

VI. Implementation

We implement PAROS both on a simulator and two real widely deployed home routers, including NETGEAR Router (R6700-AC1750) and TP-Link Router (Archer A7), using `C`. We install OpenWrt 22.03.2 on VirtualBox VM to build our simulator. We first convert openwrt.img to VBox drive. Then, we apply "uci changes" to configure OpenWrt on VM 19.07.2. The cross-compiling feature was also tested on two OpenWRT releases in both `x86` and `ARM` architecture. Both the simulator and two home routers query real-time traffic rate readings at routers every minute using `cronjobs`. We install *tcpdump* to capture network traffic rates. We write *bash* scripts to store traffic rates into *pcap* files on router ROM. We implement PAROS' algorithms and its optimizations. We deploy our PAROS' remote server on Amazon EC2 t1.micro instance with a cost of $0.0035 per hour. We store the set of artificial traffic rate signatures, indexed by time period, that are available for replay in a *SQLite*3 (version 3.39.4) database. The size of PAROS implementation is less than 2000 lines of C code.

We edit the *config* file, create *makefile*, edit the kernel module, and load the new PAROS module. This could easily be done either on a VM or home router by establishing an ssh connection. The OpenWrt device has its own local IP and MAC address, which runs just the same as a Linux machine within the Local Area Network. For user in-home activity inference attacks, we implement ML/DL-powered attack models using Keras model library [16] and TensorFlow framework [5]. Note that, we only use these attack models to benchmark different defending approaches' performance.

VII. Experimental Evaluation

A. Datasets

Dataset 1: UNSW. We downloaded publicly-available IoT traffic rate traces from UNSW Sydney [31] 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: PrivacyGuard Dataset. We downloaded PrivacyGuard dataset [38]. It was captured "mock" smart home that has 4 graduate students operating 31 IoT devices daily. PrivacyGuard was captured using NETGEAR AC1750 smart Wi-Fi router that serves as the internal switch and the gateway to the public Internet. It has totally 45 days 1 minute level traffic rate traces of 22 IoT devices.

Dataset 3: IoT Device Captures (Kaggle #1). We downloaded the IoT Device Captures dataset from Kaggle, which has 30 IoT devices and each IoT device was recorded for 40 minutes traffic rates.

Dataset 4: IoT Device Network Logs (Kaggle #2). We also downloaded the IoT Device Network Logs dataset, which captured 1 minute level network traffic traces of 14 IoT devices for 5 days using NodeMCU wifi module.

Dataset 5: Our Dataset. We also deployed 21 IoT devices in two real smart homes. The two homes are private houses that have two and three occupants, respectively. We collected second level traffic traces and their groundtruth for four weeks.

Ethical Consideration for Data Collection. We collected data to explore the severity and extent of user privacy threat from smart home network traffic traces. Our long-term goal is to provide system solutions to enable people to re-gain the control of privacy leakages. Similar to prior work, our participants were one-to-one interviewed and provided user privacy consent and agreement. We removed user identical information and sampled the datasets. *We followed our institution's Institutional Review Board (IRB) exempt process.*

B. Experimental Setup

PAROS on Virtual Machine. We implemented PAROS on a virtual machine (VM)-based simulator. This PAROS implementation covers all the major PAROS processes, including traffic rate signature learning, traffic rate injection, partial traffic padding, and optimizations.

PAROS on Router #1. We implemented and deployed PAROS on home router—NETGEAR Router (R6700-AC1750). This PAROS implementation have all the major PAROS processes, including traffic rate signature learning, traffic rate injection, partial traffic padding, and optimizations.

PAROS on Router #2. We implemented and deployed PAROS on home router—TP-Link AC1750 (Archer A7). This PAROS implementation have all the major PAROS processes, including traffic rate signature learning, traffic rate injection, partial traffic padding, and optimizations.

Attack Models. We implemented ML/DL classifiers used in prior work, including Logistic Regression, Support Vector Machines (SVMs), and Random Forest. We benchmarked different kernels for SVMs, including linear, linear passive-aggressive, linear ridge, polynomial with 1$\sim$10 degrees, and radial basis function (RBF). We also designed a convolutional neural networks (CNNs)-based DL approach to infer user activities from IoT traffic rates. Inspired by the most notable CNNs research—VGGnet [33], our CNNs architecture is comprised of input, convolutional layers (ReLU), max pooling,

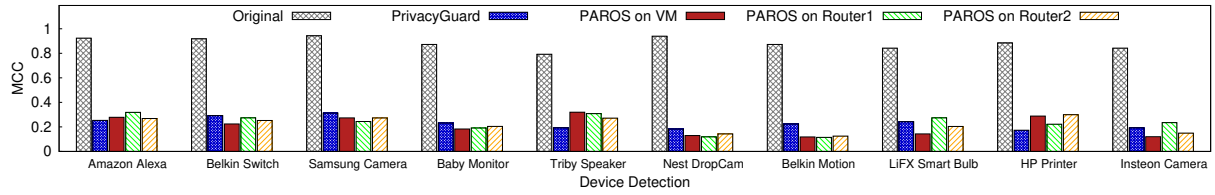

Fig. 6. Device detection performance for 10 IoT devices.

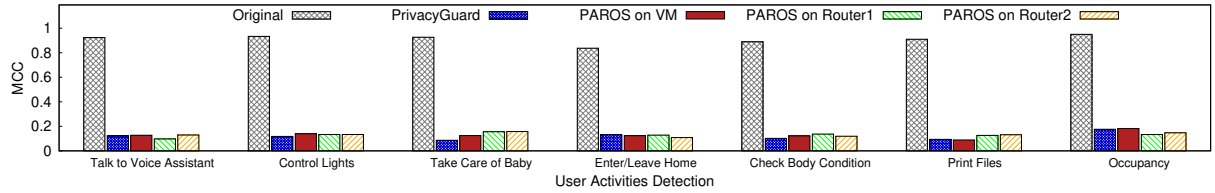

Fig. 7. User activity detection performance for 6 typical user in-home activities.

fully-connected layers and output. In addition, two fully-connected layers with ReLU and another fully-connected layer (without ReLU) are added to process the outputs.

C. Evaluation Metrics

Matthews Correlation Coefficient (MCC). We note that standard evaluating metrics (e.g, accuracy, F1) would not work well on highly imbalanced IoT traffic data [25]. We use the MCC [19], a 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 inference, 0.0 being random user activity inference, and -1.0 indicating user activity inference 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.

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

Pearson Correlation Coefficient (PCC). The PCC [24] is a measure of linear correlation between 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 correlation, 0 is no linear correlation, and -1 is total negative correlation.
Adversary Confidence (AC). We leverage AC to describe the adversary's ability to identify which time periods are corresponding to certain user activities. Given a probability p that user activity occurs independently in n time periods. AC is estimated as 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.

$$AC = \frac{np}{np + n(1-p)q} \quad (2)$$

D. Experimental Results

1) Preventing IoT Device Type Detection Attacks: We first examine the preventing ability of 4 different approaches—PrivacyGuard, PAROS on VM, PAROS on Router #1, and PAROS on Router #2 regarding IoT device type detection. Figure 6 shows the detection comparison results after applying 4 different defending approaches on 10 popular IoT devices. We report accuracy using the best performing attack model for each user activity. As we can observe from Figure 6, before applying any defending approaches, original traffic rates achieve the MCC of 0.83 across the 10 IoT devices. After applying PAROS on VM, PAROS on Router #1, and PAROS on Router #2 approaches, the modified traffic rates yield the average MCC as of 0.20, 0.22, and 0.21, respectively. The best performing attack models to detect 13 dif- ferent user activities using two datasets.Interestingly, PAROS on VM, PAROS on Router #1, and PAROS on Router #2 approaches all achieved almost the same MCC as of PrivacyGuard's—0.23. PAROS approaches achieved the same performance using significantly less hardware resources than prior work [6], [38].

Results: *PAROS approach effectively prevents a wide set of ML/DL-based IoT device type detection attacks in smart homes. In particular, PAROS yields the same low average MCC as the most notable work [38] using very limited computing resources. PAROS is the best performing privacy preserving approach that can run directly on router OS.*

2) Preventing User Activities Detection Attacks: We next examine the effectiveness of masking user activities by applying 4 different privacy preserving approaches. Figure 7 shows the detection comparison results after applying 4 different defending approaches on detecting 6 different in-home activities. As shown in Figure 7, on average, original traffic rates can achieve MCC as of 0.91. While, PrivayGuard and PAROS approaches can yield MCCs as of 0.11 and 0.12, respectively. There are slightly changes in MCCs of PAROS on VM, Router #1, and Router #2. This is mainly due the fact that the performance of PAROS on VM approach might be affected by other job scheduling load on hosted machine. In addition, Router #1, and Router #2 also slightly different levels of hardware resources in terms of CPU, RAM, ROM resources. As shown in both Figure 7 and Figure 6, PAROS approaches consistently yield the same or slightly better MCCs

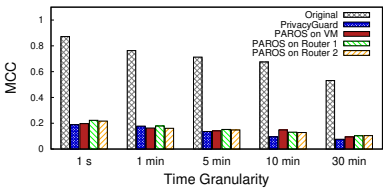

Fig. 8. Accuracy over Different Data Granularity

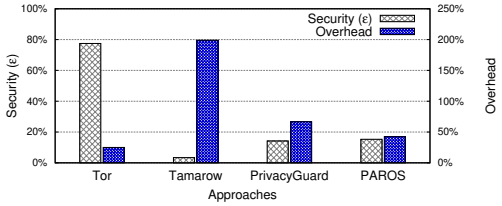

Fig. 9. Security (ϵ) and overhead comparison of different approaches.

than the most recent approach [38], which employed advance machine learning (e.g., LSTM, CNN), and other much larger scale framework to reshape traffic rates.

Results: *PAROS approach effectively prevents a wide set of ML/DL-based user activities inference attacks in smart homes. Specially, PAROS yields the same MCC as of the most notable prior work [38] using very limited computing resources.*

3) Quantifying Accuracy When Varying Granularity of Traffic Traces: We next evaluate user activity detection effect on different traffic rate traces that have different level granularities, such as 1 second, 1 minute, 5 minutes, 10 minutes, and 30 minutes. In doing so, we can examine PAROS' accuracy when attacking on different granularities traffic rate traces. As shown in Figure 8, unsurprisingly, higher granularity results in lower user activity detecting accuracy in MCC. This is mainly due to the facts that 1) PAROS approach perform consistently well on traffic rates 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 (e.g., standard deviation, variation coefficient, AUC) that machine learning-based attack models rely on will become less identifiable. This will further obscure user private information exposed in traffic rates.

Results: *PAROS' accuracy is a linear function of the granularities of traffic rate traces. PAROS yields the MCC of 0.104 when attacking on 30 minutes level traffic rate traces, which is nearly the same as random guessing, i.e., an MCC of 0.0.*

4) Quantifying Traffic Overhead: Next we will quantify the amount of network traffic overheads that are required to perform the 4 different privacy preserving approaches. Figure 9 reports the security (ϵ) and the amount of additional traffic consumption for each approach in Y1 and Y2 axis, respectively. We find that larger additional traffic overhead will result in smaller security (ϵ), which indicates smart home users can better protect their private information. Interestingly, PAROS approaches achieve the trade-off between security (ϵ) and additional traffic overhead. That says, PAROS consumes the least amount of traffic overhead while achieves the best performance to prevent user privacy leakage.

Results: *PAROS consumes the least amount of traffic overhead while achieves the best performance to protect user privacy.*

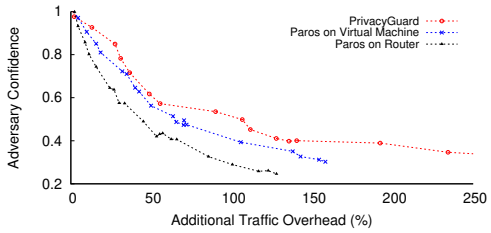

Fig. 10. Adversary Confidence for PAROS

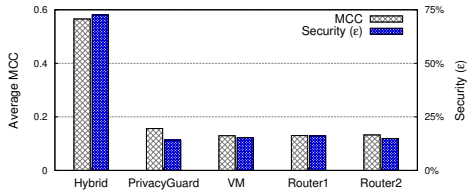

Fig. 11. The comparison on VM and Amazon best-selling home routers of 3 recent notable approaches.

5) Preventing User Activities Detection by Adaptive Adversary: We next examine the effect of different adversary confidence (AC) and different level adaptive adversary on PAROS' privacy preserving performance. As shown in Figure 10, the external attacker's AC drops significantly when PAROS is given more traffic overhead. As shown in Figure 10, PAROS (on home routers) can achieve AC as of 0.2 using 130% traffic overhead. In addition, PAROS (on home routers) coverage much faster than the most recent approach [38].

Results: *Using PAROS, the adversary's attack confidence significantly drops when user permitting additional overhead. Also, PAROS (on home routers) yields an AC of 0.2 when an adversary can consume 130% additional traffic overhead.*

6) Simulator Demonstration: Figure 11 demonstrates the performance of PAROS (simulator and two real routers), PrivacyGuard [38], and Hybrid approach. Y1 axis is reporting average MCC and Y2 axis is showing security (ϵ). We find that deploying PAROS on VM and home routers has achieved the same level effectiveness on masking user privacy. In addition to MCC and security (ϵ) of different approaches, we also examine CPU and RAM utilization for PAROS simulator and two real router deployments. As shown in Figure 12, we find that PAROS on Virtual Machine, PAROS on Router #1 and PAROS on Router #2 have the almost the same CPU and RAM utilization when reshaping traffic rates (in Mode 3) in the smart homes. For each approach, reshaping traffic rates (in Mode 3) does not increase its power consumption. That says, PAROS has demonstrated the performance consistency in the ability of protecting user private information across the simulator and two real home router deployments.

Results: *PAROS simulator yields the same system overhead as two real home router deployments. PAROS demonstrates the performance consistency in protecting user privacy in smart homes, with near-zero system overhead increasing.*

VIII. Conclusion and Future Work

We proposed a new traffic reshaping approach—PAROS that enables people to significantly reduce the private information leaked through IoT device network traffic rates. We

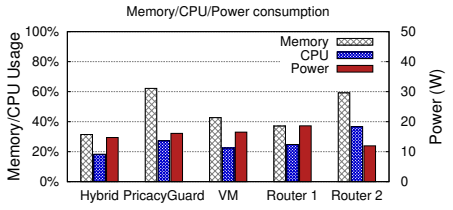

Fig. 12. The comparison of system resource usage on VM and home routers. Resource usage is calculated based on the average of three modes.(Mode 1: idling, Mode 2: PAROS Learning, Mode 3: PAROS Reshaping).

implement PAROS both simulator and new kernel service using language—C. We evaluate PAROS on open-source router OS—OpenWrt enabled virtual machine and also on two real best-selling home routers. We find that PAROS can effectively prevent a wide range of state-of-the-art adversarial machine learning-based user in-home activity inference attacks, with near-zero system overhead increasing. We plan to investigate additional efficient tiny machine learning models that might more better model user in-home activities using limited hardware resources. We will also deploy PAROS on more routers to further benchmark and improve PAROS' online performance.

Acknowledgements. This research is supported by NSF grants CNS-2238701, and the NEXUS Seed Grant.

REFERENCES

[1] Ostinato: Packet Generator and Network Traffic Generator. https://ostinato.org/, July 1th 2020.

[2] Iot Security: Hardware or Software? https://www.archonsecure.com/blog/iot-security-hardware-software, March 6th 2022.

[3] MAC address Lookup. https://www.macvendorlookup.com/api, 2022.

[4] PAROS. https://github.com/cyber-physical-systems/paros, 2023.

[5] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI'16)*, pages 265–283, 2016.

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

[7] Phuthipong Bovornkeeratiroj, Srinivasan Iyengar, Stephen Lee, David Irwin, and Prashant Shenoy. 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'20)*.

[8] T. Brewster. 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/, 2017.

[9] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, pages 227–238. ACM, 2014.

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

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

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

[13] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE symposium on security and privacy*, pages 332–346. IEEE, 2012.

[14] Md Kamrul Hasan, Husne Ara Rubaiyeat, Yong-Koo Lee, and Sungyoung Lee. A reconfigurable hmm for activity recognition. In *2008 10th International Conference on Advanced Communication Technology*, volume 1, pages 843–846. IEEE, 2008.

[15] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security*. Springer, 2016.

[16] Nikhil Ketkar. Introduction to keras. In *Deep learning with Python*, pages 97–111. Springer, 2017.

[17] Nicole Lindsey. 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/, 2019.

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

[19] Matthews Correlation Coefficient. https://en.wikipedia.org/wiki/Matthews\%_correlation\%_coefficient, Oct 2022.

[20] 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/.

[21] Rishab Nithyanand, Xiang Cai, and Rob Johnson. Glove: A bespoke website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 131–134. ACM, 2014.

[22] OpenWrt. https://openwrt.org/, Oct 2022.

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

[24] Pearson Correlation Coefficient. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient, Oct 2022.

[25] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. 2018.

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

[27] Vasanthan Raghavan, Greg Ver Steeg, Aram Galstyan, and Alexander G Tartakovsky. Modeling temporal activity patterns in dynamic social networks. *IEEE Transactions on Computational Social Systems*, 2014.

[28] Karsten Rothmeier, Nicolas Pflanzl, Joschka Hüllmann, and Mike Preuss. Prediction of player churn and disengagement based on user activity data of a freemium online strategy game. *IEEE Transactions on Games*, 2020.

[29] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

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

[31] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 2018.

[32] Statista. Internet of Things Connected Devices Installed base Worldwide from 2015 to 2025 (in billions). https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/, July 1st 2022.

[33] Very Deep Convolutional Networks for Large-scale Visual Recognition. https://www.robots.ox.ac.uk/\%~vgg/research/very_deep/, 2012.

[34] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium*, pages 143–157, 2014.

[35] Tao Wang and Ian Goldberg. On realistically attacking tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016.

[36] Tao Wang and Ian Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium*, pages 1375–1390, 2017.

[37] Wei Wang, Mehul Motani, and Vikram Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *Proc. of 15th ACM conference on Computer and communications security*, 2008.

[38] Keyang Yu, Qi Li, Dong Chen, Mohammad Rahman, and Shiqiang Wang. Privacyguard: Enhancing smart home user privacy. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (IPSN'21)*, IPSN '21, page 62–76, 2021.