Noah Apthorpe*, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster

Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping

Abstract: The proliferation of smart home Internet of Things (IoT) devices presents unprecedented challenges for preserving privacy within the home. In this paper, we demonstrate that a passive network observer (e.g., an Internet service provider) can infer private in-home activities by analyzing Internet traffic from commercially available smart home devices *even when the devices use end-to-end transport-layer encryption.* We evaluate common approaches for defending against these types of traffic analysis attacks, including firewalls, virtual private networks, and independent link padding, and find that none sufficiently conceal user activities with reasonable data overhead. We develop a new defense, "stochastic traffic padding" (STP), that makes it difficult for a passive network adversary to reliably distinguish genuine user activities from generated traffic patterns designed to look like user interactions. Our analysis provides a theoretical bound on an adversary's ability to accurately detect genuine user activities as a function of the amount of additional cover traffic generated by the defense technique.

Keywords: Internet of Things, smart home, traffic shaping, stochastic traffic padding

1 Introduction

Internet-connected consumer devices ("Internet of Things," "IoT," or "smart home" devices) have rapidly increased in popularity and availability over the past several years. Many smart home devices have always-on sensors that (1) capture users' offline activities in their living spaces and (2) transmit information about these activities outside of the home, typically to cloud services run by device manufacturers. Such communication introduces privacy concerns, not only because of the data that these devices collect and send to third parties, but also because *the very existence of traffic at all* can reveal sensitive and private information about the activities of a home's occupants.

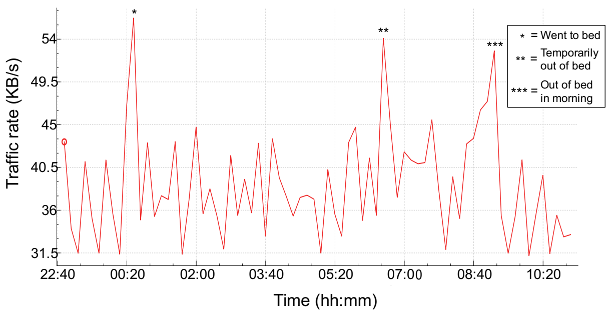

Fig. 1. Traffic rate to and from a Sense Sleep Monitor over a 12-hour period. User activities are clearly visible as traffic spikes.

In this paper, we demonstrate that despite broad adoption of transport layer encryption, smart home traffic *metadata*—specifically, traffic volumes—is sufficient for a passive network adversary to infer users' sensitive in-home activities (Figure 1). As with phone metadata [28], it is possible to learn a great deal about devices and users from Internet metadata alone. We present an attack on user privacy using metadata from smart home devices that is effective **even when devices use encryption** (Section 4). The attack involves inferring times and types of user activities from device traffic patterns.

We demonstrate this attack on commercially available smart home devices. For example, traffic rates from a Sense sleep monitor revealed consumer sleep patterns, traffic rates from a Belkin WeMo switch revealed when a physical appliance in a smart home is turned on or off, and traffic rates from a Nest Cam Indoor security camera revealed when a user is actively monitoring the camera feed or when the camera detects motion in a user's home.

The general effectiveness of this attack across smart home device types and manufacturers motivates the need for general, easy-to-deploy techniques to protect user privacy in smart homes. Conventional approaches,

***Corresponding Author: Noah Apthorpe:** Princeton University, Department of Computer Science, E-mail: apthorpe@cs.princeton.edu
Danny Yuxing Huang: Princeton University, Department of Computer Science, E-mail: yuxingh@cs.princeton.edu
Dillon Reisman: Princeton University, Department of Computer Science, E-mail: dreisman@princeton.edu
Arvind Narayanan: Princeton University, Department of Computer Science, E-mail: arvindn@cs.princeton.edu
Nick Feamster: Princeton University, Department of Computer Science, E-mail: feamster@cs.princeton.edu

such as firewalls, virtual private networks (VPN), and independent link padding (ILP), may break device functionality, provide insufficient privacy, and unacceptably increase data usage (Section 5).

In general, any privacy protection method for smart home networks involves some cost, but there is need for a tunable method that would allow users to trade off how much they are willing to spend for a guaranteed degree of privacy. To address the lack of tunable protection methods, we first define a metric, *adversary confidence*, which is the expected ratio of correct to attempted activity inferences by an adversary when traffic metadata is defended by a particular technique. We then compare adversary confidence to *bandwidth overhead*, which is the ratio of network data sent with and without a given protection technique.

We present a new traffic shaping algorithm, "stochastic traffic padding (STP)," which performs traffic shaping during user activity and injects traffic during other time periods in such a way that makes it difficult for an adversary to distinguish genuine user activity from injected traffic that mimics user activity (Section 6). With STP, the adversary's confidence in detecting a genuine activity is inversely proportional to the traffic overhead that results from the injected traffic. We demonstrate this relationship both in theory and empirically by performing STP on traffic traces from real smart home devices (Section 6.2). We also present an implementation of STP that can run on Linux-based network middleboxes, including smart home hubs, Wi-Fi access points, and home gateway routers (Section 7). In summary, our results demonstrate that just a small amount of additional traffic padding can significantly reduce adversary confidence, suggesting that STP may be practical for a wide range of deployment cases, including as a module that could run on a home network's gateway router.

We make the following contributions:

1. We demonstrate that in-home user behaviors can be inferred from traffic volumes alone, even when the traffic is protected with end-to-end encryption.
2. We evaluate conventional defenses against these inference attacks in terms of privacy protection, network delay, and traffic overhead.
3. We present stochastic traffic padding (STP), which provides sufficient privacy protection for most smart homes with considerably less bandwidth overhead than existing approaches.
4. We analyze the performance of STP using traffic traces from real devices and develop an implementation that could be used in real smart homes.

2 Threat Model

We are concerned with the ability of a passive network observer to infer users' in-home activities from smart home Internet traffic metadata. Traffic rates, network protocols, source and destination addresses, interpacket intervals, and packet sizes are accessible to many entities. Each of these potential adversaries may be incentivized to discover user behaviors, in opposition to the preferences of privacy-conscious smart home owners. We divide our threat model into two distinct classes of adversaries with differing visibility into the home network:

Local adversaries. Local adversaries include entities that can view all traffic within the smart home network. Examples of local adversaries include other malicious smart home devices, compromised or ISP-controlled home routers, and Wi-Fi eavesdroppers such as neighbors and wardrivers. Local adversaries can view MAC addresses, send times, and sizes of local Wi-Fi packets. This is the only information available to local adversaries unable to associate with the local network, such as neighbors without the WPA2 key. Local adversaries associated with the smart home network (e.g., malicious devices) can also view IP headers and transport layer headers of all packets, as well as the contents of non-encrypted DNS packets.

External adversaries. External adversaries include all entities that can view smart home traffic only after it has left the home network. External adversaries can obtain the times, sizes, IP headers, and transport layer headers of all packets leaving the smart home gateway router. External adversaries cannot view local Wi-Fi traffic within the home, and therefore cannot access device MAC addresses for identification purposes. Since most home gateway routers act as network address translators (NAT), we assume that all smart home traffic obtained by an external adversary has had source IP addresses rewritten to the single public IP of the smart home. Examples of external adversaries include ISPs, government intelligence agencies, and other on-path network observers.

Restrictions on both adversaries. We assume that packet contents are encrypted and inaccessible to the adversary. In fact, most smart home devices we tested use TLS/SSL when communicating with first and third party cloud servers. Given the increasing focus on security in the IoT community, encrypted communications will likely become standard for smart home devices. By ignoring traffic contents, our privacy attack indicates that sensitive information about user activities is still at risk even when industry best practices for data encryption are in place.

We assume that the adversary can obtain and independently evaluate smart home devices. This allows the adversary to observe device traffic patterns under various use cases. The adversary can also continuously monitor smart home traffic if it improves activity inferences. However, the adversary is not active and does not need to manipulate traffic to or from the smart home to conduct the attack.

Adversary prior knowledge. We assume that the adversary may have outside information about the activities of smart home occupants beyond what can be observed from network traffic. Adversary prior knowledge bounds both the usefulness of traffic rate analysis and the maximum effectiveness of traffic shaping defense techniques *for particular adversaries.* For example, a general adversary may know that it is more likely for any smart home occupant to go to work in the morning than in the evening. This means that smart home device traffic patterns indicating that a home occupant leaves for work in the morning are, all else being equal, more likely to be genuine than those in the evening. However, if a specific adversary, such as a neighbor, physically sees a smart home occupant leaving for work in the evening, no amount of traffic analysis or defensive traffic shaping will affect this knowledge. Nevertheless, the same defensive shaping may prevent other adversaries without this prior knowledge from inferring when the occupant leaves for work with high confidence.

Network delays. Some external adversaries may not see smart home traffic until it has traveled several hops from the home gateway router. Network delays due to congestion or other quality of service (QoS) queueing could therefore have perturbed packet timings. We disregard these perturbations because they will be insignificant relative to the timescale of user activities.

3 Experiment Setup

We set up a laboratory smart home environment with several commercially available IoT devices as a testbed for performing our traffic metadata attack and evaluating privacy protection strategies.

We configured a Raspberry Pi 3 Model B as an 802.11n wireless access point.[1] The Raspberry Pi configuration code is publicly available for research use at `https://github.com/NoahApthorpe/iot-inspector`. Wi-Fi compatible smart home devices were connected directly to the Raspberry Pi Wi-Fi network. The remaining devices were connected via Bluetooth to an Android smartphone running the latest version of the devices' corresponding applications. The smartphone was itself connected to the Raspberry Pi Wi-Fi network. This setup allowed us to record all network traffic to and from the smart home devices. We recorded traffic from the Wi-Fi interface of the Raspberry Pi to model a local adversary and from the wired interface to model an external adversary.

The commercially-available IoT devices included in our laboratory smart home are listed in Appendix Table 2. These devices are by no means exhaustive of the wide range of available IoT smart home products. However, they encompass a variety of device types, manufacturers, and privacy concerns. Given the effectiveness of traffic rate privacy attacks on all tested devices, we believe that smart home owners should be concerned about traffic rate metadata across all types of smart home devices.

4 Traffic Rate Privacy Attack

We present an attack by which a passive network observer can infer in-home user behaviors from smart home device Internet traffic metadata. The attack is applicable to nearly all currently available smart home devices and will remain an issue for new devices released in upcoming years. Without changes in developer practices or adoption of privacy protection techniques, smart home occupants will risk network observers learning about their in-home activities. Early versions of this attack [2, 3] have received press attention [8, 9, 39] for their relevance to modern IoT home devices.

4.1 General Attack Technique

As a preliminary step, the adversary identifies devices in the smart home and separates traffic metadata by device. Banner grabbing [1, 12, 14], fingerprinting [6, 35, 36], and acquisitional rule-based engines [16], have all been proposed as methods of operating system and device identification. Building on these methods, we note that external adversaries and local adversaries with network access can use DNS queries and destination IP addresses to uniquely fingerprint smart home devices, even when the devices are behind a NAT (Section 4.2.1). Local adversaries without network access

1 We followed these instructions to set up the Raspberry Pi Wi-Fi access point: `https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md`

can use the first three bytes of device MAC addresses (the organizational unique identifier) to identify device manufacturers, followed by specific device identification based on traffic rate characteristics [2].

The adversary then observes changes in traffic rates to determine the timing of user activities. Most user interactions with smart home devices occur at discrete time points or over short periods surrounded by extended periods of no interaction. For example, turning on a lightbulb, falling asleep, querying a personal assistant, and measuring blood pressure do not occur continuously. User activities generally cause noticeable changes in device traffic rates while or shortly after they occur. These changes can be brief "spikes," longer "hills", or sometimes "depressions" visible on bandwidth graphs and detectable by automated threshold methods based on standard deviations from mean traffic rates.

Finally, the adversary identifies the types of activities that cause observed traffic rate changes. The limited-purpose nature of nearly all smart home IoT devices makes this possible. Users interact with traditional computing devices, such as PCs and smartphones, for a variety of purposes, making it difficult to associate any particular change in network traffic rate with a specific activity. In comparison, once an attacker has identified the identity of a particular smart home device, it is often trivial to associate specific traffic rate changes with user activities. For example, smart outlets generally have only two functions (turning on and off) and send very little background traffic. Spikes in traffic rate at a particular time therefore clearly imply that the outlet was turned on or off at that time. Many IoT devices also exhibit very regular traffic patterns, making it easy to correlate user interactions of particular types with traffic patterns from test devices operated by an adversary.

4.2 Attack on Real Devices

We have demonstrated the effectiveness of the traffic rate privacy attack on twelve commercially available smart home devices, described as follows.

4.2.1 Device Fingerprinting & Traffic Demultiplexing

External adversaries can separate individual traffic flows and assign them to specific devices based only on destination IP addresses, even if source IP addresses and ports are rewritten by the smart home gateway NAT. All devices we tested send traffic to unique and mostly nonoverlapping sets of destination IP addresses and make corresponding sets of DNS requests (Appendix Figure 6). Additionally, many devices queried individual domains that uniquely identify the device without requiring the use of a set-based fingerprint (Appendix Table 2). For example, if an attacker sees a long-running flow to the IP address corresponding to "dropcam.com," that flow is definitely from a Nest Camera, and the traffic pattern of this flow can be used for activity inference. This ability to fingerprint based on destination IP addresses is specific to the IoT setting, because unlike web browsing or other general-purpose Internet traffic, each device sends traffic to only a few servers with which no other devices communicate, and only one flow is typically needed to perform activity inference. Destination IP address fingerprinting is also effective even if multiple device signals overlap in time or frequency, because individual packets and corresponding IP headers can still be demultiplexed. We do not consider the potential of background traffic from non-IoT devices to disrupt this fingerprinting process, because we wish to consider a worst case scenario for the defender, and because most web browsing traffic will not involve IoT device cloud servers.

Local adversaries with network access can also identify devices and demultiplex traffic using destination IP address fingerprints, but they can also use MAC addresses and other LAN-available information with a system such as Fingerbank [23] or IoT Sentinel [29].

If DNS requests or destination IP addresses are obfuscated, such as for an external adversary outside a VPN, for a local adversary without Wi-Fi network access, or through DNS over TLS (DoT) [22], DNS over HTTPS (DoH) [21], or Oblivious DNS (ODNS) [34], then device fingerprinting would have to be performed using traffic rate characteristics [2, 35]. Our STP algorithm (Section 6) protects against user behavior inference from traffic rates independent of the device identification technique employed by the adversary.

4.2.2 Activity Inference from Traffic Rates

Once traffic flows have been associated with their originating devices, the next step in the attack is to use traffic rate changes to infer user activities. We have found that activity inference is effective for all tested devices, but in the interest of space, we provide representative case studies from six devices covering a range of smart home device types:

Sense Sleep Monitor. Figure 1 shows send/receive rates from the Sense over a 12 hour period from 10:40pm to 10:40am. Notably, the send/receive rate peaked at times corresponding with user activity.

The user shut off the light in the laboratory smart home and went to bed at 12:30am, temporarily got out of bed at 6:30am, and got out of bed in the morning at 9:15am. A network observer could already guess when users sleep based on decreases in smartphones or PC web traffic; however, this assumes that smartphone and PC use only stops due to sleep, that everyone in the home sleeps at the same time and does not share devices, and that users do not leave smartphones and PCs to perform network-intensive tasks while they sleep. The single-purpose nature of the Sense sleep monitor makes none of these assumptions necessary to infer users' sleeping patterns.

Nest Security Camera. The Nest camera has at least two modes of operation: a live streaming mode and a motion detection mode. In the live streaming mode, the camera's video feed is either being actively viewed by the user through the Nest web/mobile application or being uploaded in real time to be stored on the cloud. In the motion detection mode, the video stream is not being uploaded, but the camera is monitoring the stream locally for movement. If movement is observed, the camera records a snapshot of the video and alerts the user.

Figure 2 shows send/receive rates from the Nest camera alternating between live streaming and motion detection mode every 2 minutes. The traffic rate is orders of magnitude higher in live streaming mode (and a short time afterward until the camera is notified that the user has stopped viewing the stream), allowing an adversary to easily determine whether or not the camera's live feed is being actively viewed or recorded.

Figure 2 also shows that an adversary could easily determine when a Nest camera detects movement when it is in motion detection mode. The camera was pointed at a white screen with a black square that changed location every two minutes. These simulated motion events triggered clearly observable spikes in network traffic. This predictable variability in network send/receive rates would allow an adversary to observe the presence and frequency of motion inside a smart home.

These issues are significant privacy vulnerabilities and physical security risks. It should not be possible for a third party to determine when a security camera detects movement or is being actively monitored.

Amazon Echo. We tested the Echo by asking a series of 3 questions ("what is the weather?," "what time is it?," and "what is the distance to Paris?") repeated 3 times, one question every 2 minutes. Figure 2 shows the send/receive rates of SSL traffic between the Echo and a single `amazon.com` IP address during the experiment. Although the Echo sent and received other TCP traffic to different domains during this time, we were able to identify the stream that correlated with the questions. An adversary could also identify this stream and use the SSL traffic spikes to infer when user interactions occurred.

Belkin WeMo Switch. The WeMo switch only has two states, on and off, and its network send/receive rates reflect this binarity. Figure 2 shows WeMo network behavior when the switch is turned alternatively on and off every 2 minutes using the WeMo smartphone app and the physical button on the device (both cases result in traffic to the WeMo cloud server). The spike in traffic every time the switch changes state clearly reveals user interactions with the device to an adversary.

Geeni Lightbulb. The Geeni Lightbulb also has only on and off states, and these states are reflected in network send/receive rates. Figure 2 shows Geeni Lightbulb network behavior when the bulb was turned on and off 4 times over a 37 minute period. The state changes are clearly observable as spikes in traffic rate, which could indicate when someone in a smart home turns the lights on and off. This information could in turn correlate with sleep patterns or home occupancy.

Google Home. The Google Home is able to make hands-free VoIP calls. We tested the Home by placing 5 calls to different entities over an approximately 13 minute period. Figure 2 shows the send/receive rates of traffic from the Google Home to a single `*.telephony.goog` domain during the testing period. Spikes in traffic to this domain occurred only during the start of each phone call, making detection of voice calls from the Google Home trivial for any adversary able to identify traffic to this domain, either by observing DNS requests or matching IP addresses.

5 Evaluating Existing Defenses

In this section, we evaluate three existing techniques known to prevent traffic analysis attacks in other contexts. We use two metrics, adversary confidence and bandwidth overhead, to compare the techniques. Ultimately, the ratio of adversary confidence to bandwidth overhead determines the effectiveness of the defense technique. Some techniques have a fixed ratio, but others are tunable to users' preferences for privacy versus data usage.

Adversary confidence is the expected ratio of correct activity inferences to attempted activity inferences by an adversary with no prior knowledge when

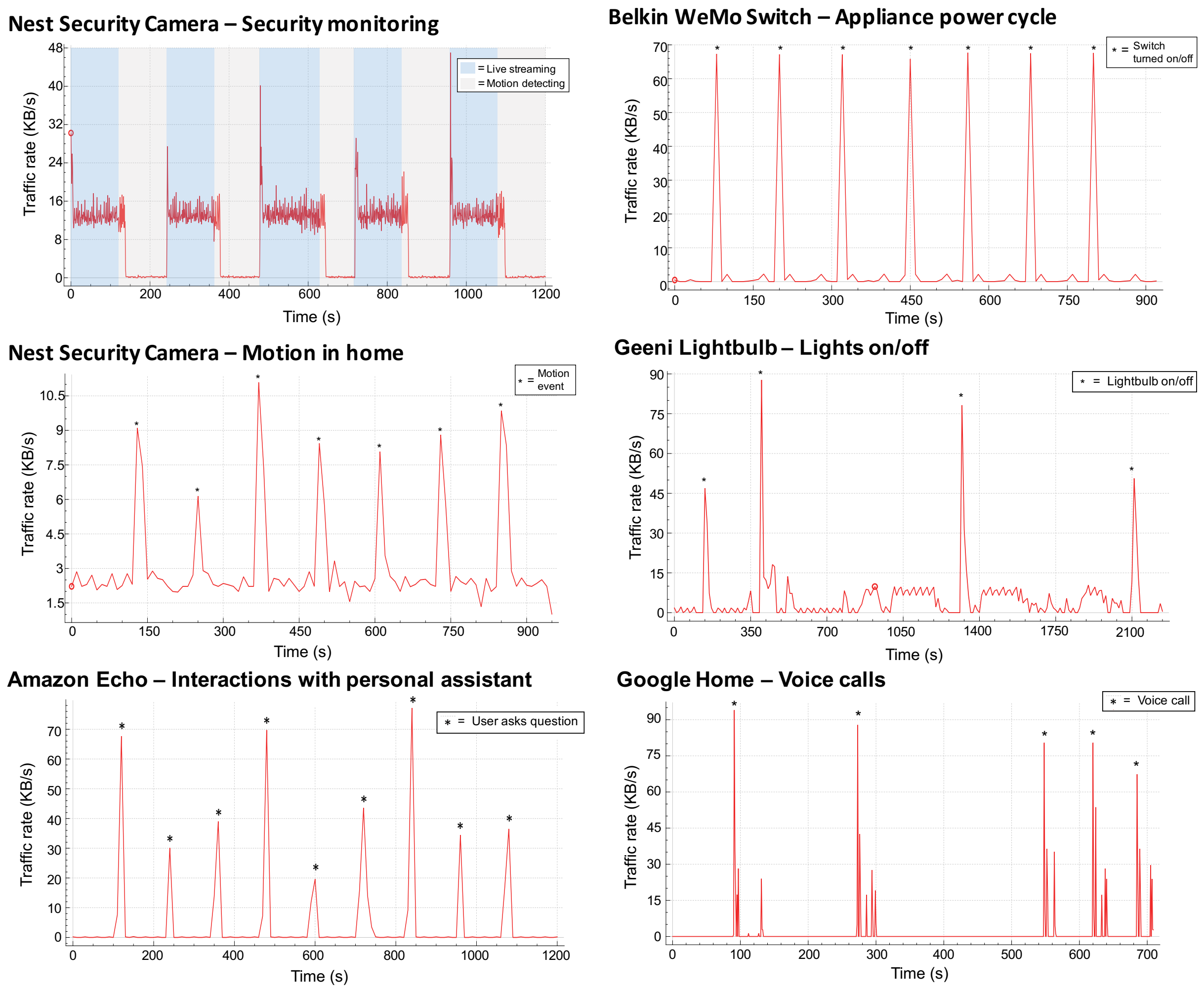

Fig. 2. Network traffic send/receive rates of selected flows from five commercially-available smart home devices during controlled experiments. Clearly visible changes in send/receive rates directly correspond with user activities. A passive network observer aware of this behavior could easily correlate smart home traffic rates with device states and corresponding user activities.

traffic rate metadata is defended by a particular technique. Lower adversary confidence means that the technique is more effective at protecting user privacy. We define c_{min} as the lowest possible activity confidence equivalent to the fraction of time that the user activity occurs. An adversary guessing that activities occur at random times will be correct this fraction of guesses. With no defense, adversary confidence ≈ 1 for all devices we tested in the previous section.

Defining adversary confidence assuming no prior knowledge makes the metric generally applicable, because it only incorporates in-band information from network traffic. Adversaries with out-of-band prior probabilities that particular user activities occur at particular times will simply combine these priors with information from traffic analysis to obtain individualized inference confidences. However, since in-band techniques will not affect these prior probabilities, the adversary confidence metric as defined captures the general effectiveness of in-band defenses for comparison.

Adversary confidence also does not rely on the computational capabilities of the attacker. This improves the generality of the metric; however, other metrics involving the cost of performing inference in different settings and for different inference algorithms may be worth exploring in future work.

Bandwidth overhead is the ratio of network data sent with and without a given defense technique. For example, a bandwidth overhead of 4 means that applying the technique results in 4 times as much traffic (e.g., in bytes) sent on the network for every user activity it protects. A lower bandwidth overhead is preferable because extra traffic contributes to network congestion and can consume user data caps.

5.1 Firewalling Traffic

The simplest technique for preventing activity inference is to prevent an adversary from collecting smart home network traffic in the first place. Configuring a firewall to block smart home device traffic is straightforward, and would protect devices that do not require WAN connectivity from user activity inference by external adversaries. However, we have found that many IoT devices do require WAN connectivity for features that users may expect involve only local communications, such as turning on an IoT outlet using a smartphone on the LAN (Appendix Table 3). Removing Internet connectivity therefore breaks the very features of these devices that incentivize consumers to purchase them instead of their non-networked counterparts. Additionally, a firewall on the home gateway router would only protect against external adversaries, leaving user activity inference by local adversaries unaffected.

Adversary confidence. Firewalling traffic results in an adversary confidence of c_{min} if the adversary is outside the firewall and an adversary confidence of ≈ 1 if the adversary is inside the firewall.

Bandwidth overhead. Firewalling traffic has a bandwidth overhead < 1, because traffic which would otherwise be sent is prevented from leaving the home.

5.2 Virtual Private Networks (VPNs)

Another technique for preventing activity inference is to tunnel all smart home traffic through a virtual private network. A VPN wraps all traffic from an endpoint in an additional transport layer, aggregating it into a single flow with the source and destination IP addresses of the VPN endpoints. This aggregation could make it difficult to determine which variations in the overall traffic rate observed from outside the VPN correspond to user interactions with individual devices.

However, the effectiveness of a VPN depends on the number of devices behind the VPN, as well as the location of the adversary and the VPN endpoints. If more devices are behind the VPN, including PCs and smartphones in addition to IoT devices, there may be more total traffic through the VPN tunnel. This may make it more difficult to de-multiplex traffic from individual devices, but the additional protection is inconsistent and difficult to quantify as individual devices may still have sparse communications and distinctive traffic patterns. Furthermore, if one VPN endpoint is on the home gateway router, then the VPN provides no protection against local adversaries. If the other VPN endpoint is visible to an external adversary (e.g., a server on an ISP's network), then the VPN also provides no protection, because the adversary can simply perform activity inference on traffic after it leaves the VPN.

We have identified three cases where an adversary can infer user activity even if the VPN is optimally located. In each of these cases, an adversary can fingerprint devices using VPN traffic rates alone:

1. Single device. If a smart home has only one device, the VPN traffic rate will match that from the device, and the attack can proceed as before.

2. Sparse activity. If there are multiple devices that send traffic at different times, time periods containing traffic from only a single device would still allow activity inference. For example, a smart door lock and smart sleep monitor are less likely to be recording user activities simultaneously. Traffic observations from particular times of day are likely to contain non-background traffic from only one of these devices. This will allow an adversary to identify the active device within a time period and perform activity inference as before. Additionally, previous work on Tor website fingerprinting indicates that machine learning techniques could potentially allow adversaries to differentiate traffic from multiple devices active simultaneously [43].

3. Dominating device. An adversary could also perform activity inference on the device that sends the most traffic if it significantly overshadows traffic from other devices. For example, traffic from a security camera uploading live video will dominate any traffic from less network-intensive devices such as smart outlets, making patterns in the camera traffic clearly observable.

Adversary confidence. As these three cases indicate, the adversary confidence provided by a VPN can range from c_{min} to 1 depending on the specific set of devices in the smart home and patterns of individual user's behavior. This adversary confidence variability motivates traffic shaping defense techniques that can guarantee certain levels of privacy protection.

Bandwidth overhead. VPNs have a bandwidth overhead of ≈ 1. A small amount of additional traffic is necessary for the creation and maintenance of the VPN tunnel, but this is negligible compared to the amount of traffic from smart home devices.

5.3 Independent Link Padding (ILP)

Independent link padding involves shaping upload and download traffic rates to match predetermined rates or schedules, thereby exposing no information about device behavior or user activities to an adversary [13, 17, 40]. Constant rate padding to enforce fixed-size packets with constant interpacket intervals is the simplest form of

ILP. An alternative method is to draw packet sizes and interpacket intervals from probability distributions independent of user behavior. Performing ILP between individual devices and cloud servers results in the following adversary confidence and bandwidth overhead.

Adversary confidence. ILP provides an optimal adversary confidence of c_{min}. No information about user activities is contained in traffic rates after ILP.

Bandwidth overhead. If the mean traffic rate without ILP is R_{normal}, the max traffic rate without ILP is R_{max}, and traffic is shaped using ILP to a mean rate R_{ILP}, the expected bandwidth overhead will be R_{ILP}/R_{normal}. However, if $R_{ILP} < R_{max}$, the device will experience network latency due to packet buffering when it attempts to send traffic faster than R_{ILP}. This latency may affect device usability, especially since devices typically require the highest send rate during user interactions.

Due to this tradeoff between bandwidth overhead and network latency, ILP can efficiently protect two specific classes of smart home devices: 1) devices with relatively constant traffic rates and 2) devices that can tolerate long network latencies. Previous work by Datta, et al. has demonstrated the effectiveness of ILP padding on smart home devices with these properties [11]. The existence of these devices is notable, because ILP padding is usually viewed as too expensive for real-world use [13].

However, many types of smart home devices have both low latency tolerance and traffic rates that spike only during user activities (Figure 2), resulting in $R_{normal} << R_{max}$ and either a high bandwidth overhead or high latency when using ILP. To verify this, we tested the performance of ILP shaping on seven devices in our laboratory smart home. We found that performing constant rate ILP on the home gateway router required approximately 40KB/s of overhead traffic in order to provide low enough latency to preserve device functionality (Appendix Table 4). This would result in 104GB of overhead data per month—excessive for smart homes with all but the highest data caps. This amount of extra traffic would also place considerable burden on ISPs if ILP achieved widespread use.

6 Stochastic Traffic Padding

We introduce a traffic shaping algorithm to defend against user activity inference from traffic rate metadata that provides an easily tunable tradeoff between adversary confidence and bandwidth overhead. This algorithm, stochastic traffic padding (STP), imposes no additional network latency and can achieve low adversary confidence for relatively little bandwidth overhead. STP can be succinctly described as follows:

1. Upload and download traffic during user activities is padded equivalently to ILP, so an adversary cannot differentiate different types of user activities.
2. Additional periods of padding are injected randomly into upload and download traffic flows. An adversary cannot distinguish these periods from real user activities, reducing confidence in activity inferences.

In the following sections, we present the STP algorithm and formalize STP's adversary confidence and bandwidth overhead (Section 6.1), evaluate STP using traffic traces from real devices (Section 6.2), describe how STP relates to categorical metadata protection (Section 6.3), and discuss how STP can adapt to complicated real-world user behavior (Section 6.4).

6.1 Algorithm and Analysis

The STP algorithm divides time into discrete periods of length T. If any privacy-sensitive user activity occurs during a time period, traffic during that period is padded to a fixed shape with mean rate R. If no user activities occur during a time period, traffic during that period is either padded to a fixed shape with mean rate R or left unmodified, depending on the output of a binary decision function. The decision function can be arbitrarily complicated in practice, but we will assume that it performs a random draw from a fixed Bernoulli distribution for the following analysis. We define a probability p that a user activity occurs independently during any time period. We could estimate this probability empirically as the fraction of time periods with traffic corresponding to user activities during a representative packet capture. We then define a probability q that the decision function chooses to perform padding during time periods without user activity.

The shape of padded traffic during a period of user activity or in response to a positive random draw is arbitrary, as long as the instantaneous traffic rate across the period is high enough that device traffic never needs to be buffered. STP implementations could shape traffic to a constant rate R, replay a short-lived flow scaled to mean rate R, or choose any other fixed traffic shape with mean rate R. We use constant rate padding in the following sections to simplify visualizations. If the maximum rate of device traffic is known from traffic traces, R can be set directly. Otherwise, R can be started at a high value and periodically decreased as long as it remains higher than any previous traffic rate during user activity. Keeping R slightly above this maximum allows

STP to act with minimal network latency and bandwidth overhead.

Bidirectional traffic. Most smart home devices use bidirectional protocols, especially TCP and HTTP, to communicate with cloud servers. This means that user activities may be reflected in the patterns of both upload and download traffic. STP therefore pads traffic in both upload and download directions for the entirety of each time period containing user activity or selected by random draw. Insofar as the time period length T is long enough to cover complete bidirectional communications (requests and responses) corresponding to user activities, all periods of padding will be indistinguishable in both directions. For example, suppose an IoT device sends a request and receives a response in a single TCP connection with the SYN packet at T_{SYN} and FIN packet at T_{FIN}. The bidirectional STP traffic will start at T_{SYN} and continue for T, such that $T_{FIN} - T_{SYN} \leq T$. For DNS and other known UDP protocols, T will be made long enough to overlap the request and response packets given the latency and bandwidth of the network.

Shaping bidirectional traffic with STP involves two additional considerations. First, one direction may have a considerably higher volume of traffic, such as short HTTP GET requests versus longer HTTP responses. In this case, using the same shaped mean rate R in both directions would be wasteful, as one direction requires substantially less cover traffic to mask user activities. Instead, STP can use separate shaped traffic patterns with mean rates R_u and R_d for the upload and download directions, respectively. Choosing R_u and R_d can be performed as described for R above. For the following analyses, we define $R = R_u + R_d$ to reason about the total overhead of STP shaping in both directions.

Second, STP shaping must be applied to upload and download traffic at different locations in the network. Upload traffic could be shaped on the devices themselves or by a middlebox in the smart home, such as an IoT hub or gateway router. Download traffic could be shaped on the cloud servers or by a middlebox in the network, such as a VPN end point. These locations must communicate to synchronize periods of padding. This communication could occur through the contents of encrypted cover traffic, but is implementation dependent. Deciding on which locations to implement STP for bidirectional traffic determines whether it protects against both internal and external adversaries or external adversaries only (Section 7). The following analysis assumes that the adversary can only see traffic shaped by STP.

Adversary confidence. The adversary's goal is to decide which periods of padding correspond to user activities. We can bound adversary confidence and bandwidth overhead based on the frequency of user activities and the probability of non-activity padding. Over any given set of n time periods, the expected number with user activities is np. However, the expected number of padded periods after STP is $np + n(1-p)q$. Since the adversary cannot tell these periods apart, the expected adversary confidence c is

$$c = \frac{np}{np + n(1-p)q} = \left(1 + \frac{(1-p)q}{p}\right)^{-1} \tag{1}$$

This is a simple power law that intuitively matches the behavior of STP. If user activity occurs more frequently (higher p), any particular period of padding is more likely to correspond to a user activity. If non-activity padding occurs more frequently (higher q), any particular period of padding is less likely to correspond to a user activity.

An adversary continuously performing inference attacks against STP (e.g., an ISP with a permanent tap on a smart home's WAN traffic) will eventually "succeed," in the sense that c fraction of all inferences claiming that user activity occurred during a particular time period will be correct in expectation. However, the adversary will not know *which* of these inferences are correct, gaining no useful information for low c.

Bandwidth overhead. In order to quantify bandwidth overhead, we define two additional quantities, D_A and $D_{\neg A}$, the average amount of bidirectional data sent during periods of user activity and periods of background traffic, respectively, before STP. The expected average bandwidth overhead b of STP is

$$b = \frac{pRT + (1-p)qRT + (1-p)(1-q)D_{\neg A}}{pD_A + (1-p)D_{\neg A}} \tag{2}$$

The values of p, D_A, $D_{\neg A}$, and to some extent R are determined by users and device developers, but the STP algorithm can adjust q to trade off adversary confidence and bandwidth overhead. Considering the limits, $q = 0$ means that only periods of real user activities are subject to padding, so the adversary can be certain that user activity occurred during these periods. $q = 0$ also has the lowest bandwidth overhead, because no cover traffic is sent during periods of no activity. In comparison, $q = 1$ is equivalent to ILP as described in the previous section. $q = 1$ has the highest bandwidth overhead because it requires constant cover traffic.

Privacy and overhead tradeoff. By increasing q, adversary confidence decreases according to a power law

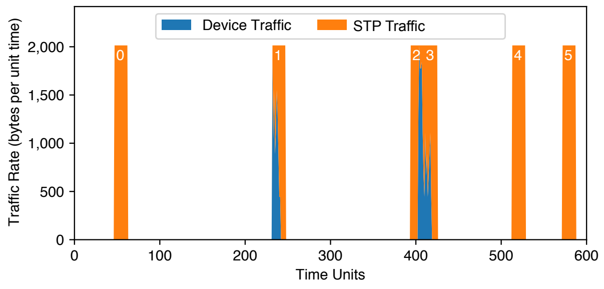

Fig. 3. Example of STP applied to smart switch traffic.

while bandwidth overhead increases linearly. The ratio of adversary confidence to bandwidth overhead is also a power law:

$$\frac{c}{b} = O\left(q^{-2}\right) \tag{3}$$

This allows STP to choose a value q at the point of maximum curvature in c/b to provide the optimal adversary confidence and bandwidth overhead tradeoff (given no outside criteria).

To visualize how c and b change as we vary p and q, we plot Figure 4(a) based on Equations 1 and 2. The horizontal axis shows the bandwidth overhead, b, and the vertical axis shows the adversary confidence, c. There are two curves, presented in two different colors, which correspond to $p = 0.01$ and $p = 0.1$, respectively. Each curve has exactly 101 data points. The leftmost point corresponds to $q = 0$. The rightmost point corresponds to $q = 1$. We set $RT = 1$, $D_A = 0.9$, and $D_{\neg A} = 0$. These values mean that there is no background traffic, 0.9 units of data are sent per period with user activity, and 1 unit of data is sent per period of padding.

Regardless of p, both curves follow a similar shape that is typical of power-law graphs. As q increases from 0, the decrease in c is initially drastic but flattens as q approaches 1. This suggests that STP with even a small bandwidth overhead can significantly reduce adversary confidence and improve privacy. However, when q approaches 1, significantly more cover traffic is needed to reduce the adversary confidence by the same amount.

Two points on the chart are of particular interest. The lower-right point where $q = 1$ denotes the case where we are effectively doing constant-rate ILP. The adversary confidence reaches $c_{min} = p$. In contrast, the upper-left point where $q = 0$ represents the case where the STP traffic occurs only during a user activity. Here, an adversary can trivially infer when a user activity takes place, i.e., $c = 100\%$.

6.2 Evaluation with Device Traffic

While we can use the theoretical model to demonstrate the general power-law relation between c and b, it is difficult to show exactly how c and b vary with p and q for real devices. User activity traffic is likely to vary across devices in both duration and bandwidth of each activity, as we have shown in Figure 2.

To evaluate our model while accounting for these device-specific differences, we follow a trace-driven approach. Specifically, we construct a simulator that generates traffic in a two-step process. It first generates traffic from user activities at different values of p based on packet traces that we have collected from real devices. The simulator then applies STP to this generated traffic at different values of q. We measure c and b over time for varying p and q.

Generating user activity traffic. We first analyze the packet traces from actual devices (obtained in Section 4.2) and extract the duration and bandwidth associated with each user activity. These packet traces include traffic from 7 WeMo switch user activities, 9 Amazon Echo activities, and 7 Nest security camera activities. WeMo switch activities last an average of 1 second, Echo activities last 2 to 5 seconds, and Nest camera activities last 8 to 15 seconds.

We create one simulator for each device. At time t, a device simulator makes a binary decision whether or not to simulate an activity with a probability p. If the simulator decides to simulate an activity at t, then it randomly picks an activity traffic trace recorded from the device and replays it in the simulation. If the replayed traffic lasts t' seconds, the simulator will not decide whether to simulate another activity until time $(t + t')$. This creates a traffic trace with realistic activity traffic patterns that occur according to a Bernoulli process with tunable probability p. For simplicity, we assume there is no background traffic (i.e., $D_{\neg A} = 0$).

Generating STP traffic. Our simulator next applies STP to the user activity traffic generated in the first step. The simulator knows the set of replayed user activities, allowing it to set R and T higher than the maximum traffic rate and duration of any user activity. The behavior of the simulator at time t follows one of two cases:

Activity padding. If the simulator sees the first byte of traffic from a user activity, it instantaneously starts padding with duration T, regardless of the value of q. All user activities are therefore masked with padding, which an adversary cannot distinguish from padded periods that do not cover user activities.

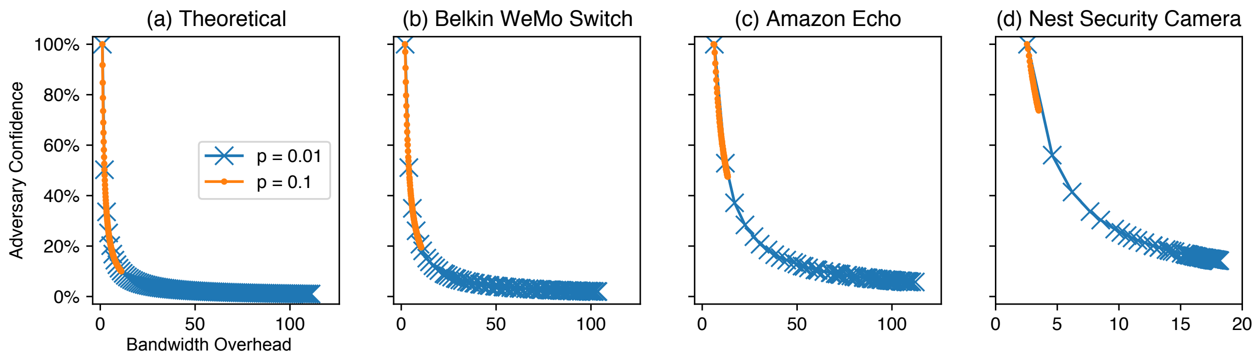

Fig. 4. STP trade-off between bandwidth overhead and adversary confidence for different devices and user activity frequencies, p. Each point corresponds to a probability of non-activity padding, q, ranging from 0 (highest adversary confidence) to 1 (highest bandwidth overhead) in steps of 0.01. Note the inverse square relationship, which allows STP to achieve low adversary confidence for relatively little bandwidth overhead.

Non-activity padding. If the simulator currently observes no user activity traffic, then it makes a binary decision whether or not to perform traffic padding. The decision follows a Bernoulli process with a probability q. If the simulator decides to pad, the padding will have duration T and traffic rate R. If no user activity occurs, the simulator will not make another decision until time $(t + T)$. However, if user activity occurs before $t + T$, the simulator adjusts the cover traffic rate based on the device traffic rate, so that the total traffic rate remains R. If the user activity continues past $t + T$, the simulator continues padding until $t + 2T$. This prevents an adversary from distinguishing a user activity that overlaps a time period boundary from any other two periods of back-to-back padding.

Visualizing STP traffic. To illustrate how STP affects device traffic, we run the simulator based on the WeMo switch with $p = 0.001$ and $q = 0.01$. Figure 3 shows the resulting traffic over 6 activities. Areas 0, 4, and 5 represent non-activity padding. Area 1 represents activity padding. Areas 2 and 3 represent an activity that starts during non-activity padding and overlaps $t + T$. For this particular traffic pattern, an adversary has a $c = 2/6 = 33.3\%$ chance of guessing the time of user activities with a bandwidth overhead of $b = 6.8$.

Visualizing STP trade-offs. Figure 4(b–d) shows the trade-offs between c and b for the three device simulators and varying p and q. For each device, we vary p from 0.01 to 0.1 and increase q from 0 to 1 at increments of 0.01. We run the simulation over 10,000 seconds. We repeat each run 50 times and plot the mean values of c and b.

Similar to the theoretical result in Figure 4(a), all three curves follow the typical power-law shape, which shows increasing bandwidth overhead to achieve the same reduction in adversary confidence as q increases. However, there are three notable differences across devices.

Bandwidth overhead at $q = 0$. Even at $q = 0$, there is still bandwidth overhead due to padding during user activities. For the WeMo switch and the Nest camera, the overhead at $q = 0$ and $p = 0.01$ is 2.0 and 2.6 respectively, while the overhead is 6.4 for the Amazon Echo. This difference is due to the variations in user activity traffic rates across devices. The recorded user activities from the WeMo switch have a standard deviation in traffic rate that is 4.7% of the mean. This is compared to 28.7% for Nest camera activities and 59.2% for Amazon Echo activities. As R and T are set based on the maximum duration and bandwidth of user activity traffic for a specific device, the bandwidth overhead when padding user activities is higher when there are more variations in activity traffic rates.

Curve slope. The bandwidth overhead required to reduce adversary confidence by the same amount is different across devices. For the WeMo switch, increasing q from 0 to 0.01 raises the bandwidth overhead from 2.0 to 4.0 and reduces adversary confidence from 100.0% to 51.1%. Effectively, 0.04 extra bandwidth overhead is needed to achieve a one percentage point decrease in adversary confidence. In contrast, 0.05 and 0.12 extra bandwidth overhead is needed to achieve the same confidence decrease for the Nest camera and Amazon Echo, respectively. The Amazon Echo's extra bandwidth overhead is the highest, because its user activity traffic has the most variations and thus requires the most cover traffic per user activity.

Bandwidth overhead and adversary confidence at $q = 1$. Bandwidth overhead at $q = 1$ is the lowest for the Nest camera ($d = 18.2$). User activities on the camera also have the longest duration; as such, $q = 1$ results in the least duration of non-activity padding. Furthermore, since the duration of injected traffic is the least among the devices, the adversary confidence is the highest at $q = 1$ for the camera ($c_{min} = 14.2\%$).

Summary. Using a trace-driven approach, we have constructed three STP simulators for the WeMo switch, the Nest camera, and Amazon Echo. We have shown that STP bandwidth overhead has an inverse-square relation with adversary confidence for the p and q values we tested. Moreover, we have demonstrated that the exact trade-offs between overhead and adversary confidence differ across the devices. Other devices with similar user activity traffic as the three simulated devices are likely to exhibit similar trade-offs. Table 1 compares STP to the techniques discussed in Section 5.

6.3 Obfuscating Categorical Metadata

STP is focused on protecting *traffic rate* metadata (packet times and sizes) from user activity inference. However, network traffic flows also contain *categorical* metadata, such as protocols, DNS hostnames, and IP addresses, that could leak information about user activities. For example, we were able to use DNS hostnames to identify smart home devices (Section 4.2.1). DNS queries to specific third-party services (e.g., from an Amazon Echo to a music streaming platform [3]) could also directly indicate user interactions.

In order to completely protect a smart home from activity inference, STP must be combined with a method to remove or obfuscate categorical metadata related to user activities. Fortunately, methods to protect categorical metadata of network traffic are widely available. The cleanest method is to tunnel all STP traffic through a VPN, which groups all smart home traffic (including DNS) into a single flow with a single protocol and packet header information uncorrelated with user behaviors. Our middlebox STP implementation (Section 7.1) uses this approach. While requiring a VPN raises threshold for adoption, personal VPN usage is becoming more common due to increased privacy awareness [18] and availability of non-enterprise VPN services, such as home routers with built-in VPN [33] and Google's Project Fi VPN [25]. Alternatively, specific categorical metadata could also be protected by existing protocols, such as obfuscating DNS via DNS over TLS (DoT) [22], DNS over HTTPS (DoH) [21], or Oblivious DNS [34]. In our presentation of STP, we therefore focus entirely on traffic rate metadata, assuming that sensitive categorical data can be removed from smart home traffic by alternative methods.

6.4 Adapting to Real-World User Behavior

Real-world user behaviors are often more complicated than the simple Bernoulli model assumed in Section 6.1. While this assumption simplifies activity confidence and bandwidth overhead derivations, it does not limit the generality of STP. Instead, it highlights where the algorithm could be extended to handle more nuanced user activity patterns that occur in practice.

Activity correlations. Our derivations of adversary confidence and bandwidth overhead assume that user activities are independent and Bernoulli distributed. This prevents an attacker from using inter-activity intervals, the amount of time between padded periods, to help distinguish user activities. For many smart home devices, this assumption holds. Just because a user turns on a lightbulb at a particular time doesn't provide any information *a priori* about when the lightbulb will be turned off. Users may query a personal assistant frequently or infrequently with no apparent pattern. However, other devices, such as a washer with distinct cycles, may have priors on the temporal spacing of user activities that could help an adversary distinguish activity and non-activity padding. Packets from unknown bidirectional UDP protocols may also exhibit long-term temporal correlations unknown *a priori.*

However, STP is still effective even if user activities exhibit temporal patterns. The decision function can use a temporal or causal model instead of a Bernoulli distribution to choose when to trigger non-activity padding. We have found that hidden Markov models can be trained to mimic the patterns of real user activities (Figure 5). Such models could be used to dictate realistic timings of non-activity padding intervals. An STP implementation could initially perform ILP (e.g., constant rate padding) while collecting training data from user activity patterns then switch to STP once a better model has been trained. This model can be refined online as more user data is observed. Of course, even the best models could be fooled by long-term recordings and/or external information, as real user behaviors are driven by variables unavailable to an on-path network device. Ultimately, STP represents a tradeoff between privacy and additional traffic volume.

Long user activities. Some devices may involve user activities of widely variable or unbounded length. For example, a user watching a live video feed from a security camera may check the feed for a few seconds from a smartphone or may leave the feed open in a browser tab for an entire afternoon. STP still works in such cases, but we have to relax the assumption that

Defense	Adversary Confidence	Bandwidth Overhead
Firewalling Traffic	≈ 1 inside firewall, c_{min} outside firewall	< 1
Virtual Private Network (VPN)	Varies from c_{min} to 1 depending on devices & user behaviors	≈ 1
Independent Link Padding (ILP)	c_{min}	R_{ILP}/R_{normal}
Stochastic Traffic Padding (STP)	Tunable from ≈ 1 to c_{min} with $O(q^{-1})$	Varies with $O(q)$

Table 1. Comparison of defenses against in-home activity inference from smart home traffic rate metadata.

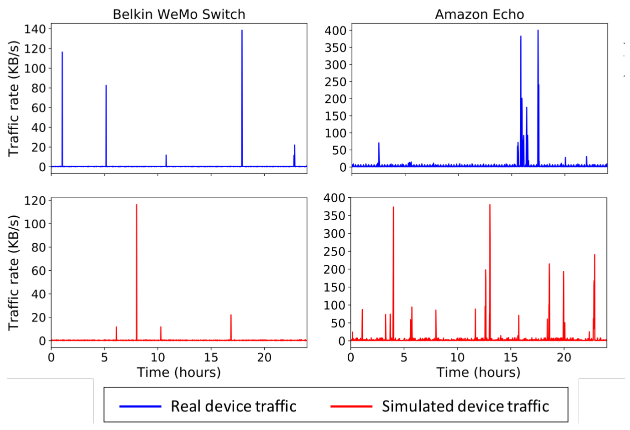

Fig. 5. The start times of spikes in IoT device traffic traces simulated by hidden Markov models could be used to dictate realistic timings of non-activity padding intervals in STP.

all user activities fit into one time period. Instead, user activities may span multiple time periods, all padded. Non-activity padding may also span multiple time periods. The durations of non-activity padding must be chosen such that they are statistically indistinguishable from the distribution of real activity durations. This could be performed by fitting a model to the distribution of user activity durations and using the model to generate non-activity padding durations. This model could be continuously refined as more user data is observed.

Cross-device correlations. The proposed STP algorithm also assumes that there are no correlations between user activities across different devices. Such correlations would not be present for non-activity padding periods, making it possible for an adversary to distinguish the times of user activities. For example, consider a home with a smart washer and a smart dryer. It is unlikely that the dryer will be run before the washer. Even if washer and dryer activities separately meet all of the above assumptions, an attacker would still be able to identify some non-activity padding periods by comparing across devices.

Information leakage to external adversaries from cross-device correlations could be prevented by performing STP at the level of an entire smart home instead individually for each device. Traffic from all devices would be merged into a single flow (e.g. over a VPN from a home gateway router) and then STP could be performed treating the entire home as a single "device." Our STP implementation has this behavior (Section 7.1).

7 STP Implementation

In this section, we describe two ways to implement STP: on middleboxes (e.g., home routers) and on IoT devices.

7.1 Implemented on Middleboxes

We have created a service that enables STP on any Linux-based network middlebox, such as a smart home hub, Wi-Fi access point, or home gateway router. The service has been tested on the Raspberry Pi Wi-Fi access point in our laboratory smart home and consists of two components: a Python script that performs traffic shaping and a custom VPN endpoint.

Traffic shaping. The traffic shaping script contains logic to decide when to perform periods of constant-rate padding for STP. The padding itself is implemented using the Linux kernel's traffic control system, configurable via the `tc` tool, combined with a user-space program that generates cover packets (Appendix Figure 7). The script treats all traffic through the middlebox as one "device" (although MAC address filters can be specified to exclude PCs, smartphones, or other non-IoT devices). Otherwise, the script logic matches the STP algorithm presented in Section 6.1. Default threshold-based activity detectors and Bernoulli decision functions can be parameterized or replaced with more complicated versions if desired.

VPN. The traffic shaping script automatically connects to an OpenVPN instance hosted on Amazon EC2. Both VPN endpoints communicate to pad traffic in the upload and download direction, protecting bidirectional protocols as described in Section 6.1. Both VPN endpoints also automatically drop cover packets to prevent them from confusing devices or cloud servers.

Protection. Performing STP on a middlebox, either a smart home hub or a home gateway router, protects against activity inference by external adversaries, but not from local adversaries with access to device Wi-Fi traffic.

7.2 Implemented On Devices and Servers

Device developers could alternatively include STP as a feature of their devices, either as custom code or a third-party library. STP shaping of upload traffic would occur on device, while shaping of download traffic would occur on cloud servers. This implementation approach would require the least user effort. It would also allow developers to specify distributions of activity lengths or inter-activity intervals for STP based on device implementation details or the space of possible user interactions. Performing traffic shaping on devices would protect first-hop Wi-Fi traffic, preventing both local and external adversaries from performing activity inference. This is important because many users may be more concerned about details of their in-home behaviors leaking to nearby adversaries (nosy neighbors, potential burglers, etc.) than to their ISP. However, the cover traffic required by STP will place increased burden on device manufacturers' cloud infrastructure. Although this overhead is far less than would be required for ILP shaping, device developers will likely be unincentivized to include STP on devices without a considerable increase in consumer concern about metadata privacy risks. In the meantime, traffic shaping at network middleboxes remains the most viable option for privacy conscious smart home device owners.

8 Future Work

Future research efforts could further explore the threat of the activity inference attack and continue to improve STP and related defense techniques.

Fine-grained and higher-order user activity inference. The attack we describe involves mostly binary inferences (device state changes) from traffic rate metadata from individual smart home devices. We are next interested in whether combining traffic rate metadata with categorical and physical layer metadata (e.g., WiFi radio signal strengths) can reveal finer-grained user interactions, such as what smart TV channel a user is watching. We are also interested in whether combining metadata from multiple devices allows for inference of higher-order user behaviors, such as "hosting a party" or "late night working." Fine-grained and higher-order activity inferences may both constitute privacy violations, and it would be beneficial if researches could warn consumers about these risks.

Active adversaries. Future work could also consider user activity inference by active attackers. Unlike undetectable passive surveillance, active adversaries could break STP by interrupting traffic flows to determine which periods of padding correspond to actual user activities. For example, an active adversary could drop packets during periods of padding. If this is followed by an unusually high frequency of padded time periods, it may indicate that the dropped packets contained actual device traffic and the device is trying to troubleshoot the loss of connectivity. Otherwise, the dropped packets were likely cover traffic.

Reducing STP bandwidth overhead. STP shapes traffic to fixed patterns chosen to cover all possible flows. This means that relatively low-volume flows from user activities could require large amounts of cover traffic to match these fixed patterns. Future work could further reduce the bandwidth overhead of STP by allowing more fine-tuning of shaped traffic to account for low bandwidth "mice" and high-bandwidth "elephant" flows. Rather than R_u and R_d, STP implementations could have more options for shaped traffic patterns with mean rates $[R_0, R_1, \ldots]$. Time periods with real user activity would be padded to the pattern with the minimum R_x that still covers the device traffic. This is reminiscent of defenses against website fingerprinting proposed by Nithyanand et al. [31] and Wang et al. [42], which shape finite length traffic flows to match supersequences over anonymity sets of packet sequences.

However, care would have to be taken when choosing which patterns to use for padded time periods without user activity. The frequency and timing of each pattern could create a new channel that leaks information about the likelihood of these periods containing only cover traffic. Combining multiple shaped traffic patterns with the real-world considerations discussed in Section 6.4 would introduce additional multi-variable relationships to STP, complicating adversary confidence and bandwidth overhead derivations into a topic for future work.

9 Related Work

This paper draws on a rich history of related research on traffic analysis attacks and prevention techniques. The attack we describe is similar in spirit to the Fingerprint and Timing-based Snooping (FATS) attack presented by Srinivasan et al. in 2008 [38]. The FATS attack involves activity detection, room classification, sensor classification, and activity recognition from Wi-Fi traffic metadata from a sensor network deployed in the home, the precursor to today's smart home IoT devices. In contrast to our attack, FATS relies on radio fingerprinting and signal attenuation measurements that are not available to external adversaries.

Other research has demonstrated traffic analysis attacks on specific IoT devices [2, 3, 20]. Copos et al. [10] used metadata to detect transitions between Home and Auto Away modes of Nest Thermostat and Nest Protect devices. Our work re-emphasizes metadata privacy concerns demonstrated by these projects for a broader range of modern smart home devices.

Our attack also draws from side-channel privacy attacks using network traffic metadata on anonymity networks [4, 30], Internet browsing patterns [15, 19], and user/device fingerprinting [5, 26, 41].

Similarly, our development of STP was motivated by existing work on traffic shaping for privacy. Park et al. have described "activity cloaking," a technique related to STP which is designed to protect against the FATS attack [32]. Activity cloaking involves some devices generating fake data to mimic actual private activities. This is similar in motivation to STP, but has several important distinctions. Activity cloaking doesn't shape traffic from real activities, requires participation of many devices (and corresponding adoption by many devices/companies), is focused on Wi-Fi eavesdroppers rather than WAN observers, and cannot be applied solely on a gateway router or other middlebox.

Liu et al. have described a community-based differential privacy framework to protect smart homes against traffic analysis [27]. Their approach involves sending traffic between the gateway routers of multiple cooperating smart homes before forwarding it to the Internet. This obfuscates the originating home of the traffic with minimal bandwidth overhead. However, this approach could result in long network latencies if the homes are not geographically proximal, and the requirement that multiple homes cooperate raises the bar for adoption.

Finally, STP was motivated by research on traffic shaping to prevent website fingerprinting and flow correlation in anonymity networks, primarily Tor. This includes independent link padding algorithms [17, 40], such as BuFLO [13], which force traffic to match a predefined schedule or distribution independent of the unshaped traffic, and dependent link padding algorithms, in which unshaped traffic patterns affect the shaped output. Independent link padding algorithms are effective at preventing user activity inference (Section 5.3). However, most recent dependent link padding algorithms cannot protect against user activity inference.

Ultimately, we cannot use one of these techniques instead of STP, because defending against website fingerprinting is a fundamentally different problem with quite different assumptions and goals. Website fingerprinting involves comparisons *between* traffic traces (e.g. "Is this trace similar to previous traces known to be from fetching a particular website?") while user activity inference involves analysis of patterns *within a single trace* (e.g., "Is this traffic spike substantively higher than the background, suggesting a user interaction has occurred?"). In other words, website fingerprinting asks "which website is being fetched?" while user activity inference need merely ask "is any website being fetched?" Any use of a single-purpose smart home device may be something a user considers private.

Dependent link padding algorithms include adaptive padding, proposed by Shmatikov and Wang, which forces inter-packet intervals of short-lived web communications through an anonymity network node to match a pre-specified probability distribution [37]. Wang et al. also designed an algorithm that uses matched packet schedules to prevent an observer of an anonymity network mix node from pairing incoming flows with outgoing flows [45]. In 2016, Juarez et al. applied adaptive padding to prevent Tor website fingerprinting (WTF-PAD) [24]. Cai et al. also presented a defense against Tor website fingerprinting (Tamaraw) that shapes website downloads to multiples of a padding parameter L packets [7]. Wang and Goldberg have also proposed a defense against website fingerprinting (Walkie-Talkie) that uses half-duplex communication to limit the information available to the adversary [44].

All of these dependent link padding techniques allow periods of higher or lower traffic rates to be preserved in the shaped output, as long as the traffic is smoothed to be indistinguishable from traces from other websites. However, unlike STP, none of these techniques introduce spikes or other periods of high traffic rates during device (or browser) quiescence to confuse adversaries about when user activities occur. Applying these techniques to smart home traffic would still allow an adversary to perform user activity inference, because fluctuations in shaped traffic rates would still be likely correlated with user interactions.

10 Conclusion

The privacy threat from traffic metadata will continue to grow along with the market for IoT smart home devices. In this paper, we show that a passive network adversary can infer private in-home user activities from smart home traffic rates *even when devices use encryption.* We introduce "stochastic traffic padding"

(STP), a new traffic shaping algorithm which uses intermittent periods of traffic padding to limit an adversary's confidence that times and types of user activities inferred from traffic correspond to real user behaviors. STP provides a tunable tradeoff between adversary confidence and bandwidth overhead, allowing sufficient privacy protection without significantly decreasing network performance or consuming data caps. We demonstrate the effectiveness of STP on traffic traces from real smart home devices and present an implementation for smart home hubs, Wi-Fi APs, and gateway routers.

References

[1] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., et al. Understanding the mirai botnet. In *USENIX Security Symposium* (2017), pp. 1092–1110.

[2] Apthorpe, N., Reisman, D., and Feamster, N. Closing the blinds: Four strategies for protecting smart home privacy from network observers. *Workshop on Technology and Consumer Protection (ConPro)* (2017).

[3] Apthorpe, N., Reisman, D., and Feamster, N. A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic. *Data and Algorithmic Transparency Workshop (DAT)* (2017).

[4] Back, A., Möller, U., and Stiglic, A. Traffic analysis attacks and trade-offs in anonymity providing systems. In *International Workshop on Information Hiding* (2001), Springer, pp. 245–257.

[5] Bellovin, S. M. A technique for counting natted hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment* (2002), ACM, pp. 267–272.

[6] Caballero, J., Venkataraman, S., Poosankam, P., Kang, M. G., Song, D., and Blum, A. Fig: Automatic fingerprint generation. In *Network and Distributed System Security Symposium* (2007).

[7] Cai, X., Nithyanand, R., Wang, T., Johnson, R., and Goldberg, I. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 227–238.

[8] Chirgwin, R. Internet of snitches: anyone who can sniff 'thing' traffic knows what you're doing. `https://www.theregister.co.uk/2017/05/29/internet_of_snitches_anyone_who_can_get_your_traffic_knows_what_youre_doing/`, May 2017.

[9] Coldewey, D. Internet providers could easily snoop on your smart home. https://techcrunch.com/2017/08/28/study-tracks-what-smart-home-activity-can-be-seen-by-internet-providers/, August 2017.

[10] Copos, B., Levitt, K., Bishop, M., and Rowe, J. Is anybody home? inferring activity from smart home network traffic. In *Security and Privacy Workshops (SPW), 2016 IEEE* (2016), IEEE, pp. 245–251.

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

[12] Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., and Halderman, J. A. A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), ACM, pp. 542–553.

[13] Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Security and Privacy (SP), 2012 IEEE Symposium on* (2012), IEEE, pp. 332–346.

[14] Fachkha, C., Bou-Harb, E., Keliris, A., Memon, N., and Ahamad, M. Internet-scale probing of cps: Inference, characterization and orchestration analysis.

[15] Felten, E. W., and Schneider, M. A. Timing attacks on web privacy. In *Proceedings of the 7th ACM conference on Computer and communications security* (2000), ACM, pp. 25–32.

[16] Feng, X., Li, Q., Wang, H., and Sun, L. Acquisitional rule-based engine for discovering internet-of-things devices. In *27th {USENIX} Security Symposium ({USENIX} Security 18)* (2018), pp. 327–341.

[17] Fu, X., Graham, B., Bettati, R., Zhao, W., and Xuan, D. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Parallel Processing, 2003. Proceedings. 2003 International Conference on* (2003), IEEE, pp. 483–492.

[18] Gargiulo, M. The future of the VPN market. `https://www.forbes.com/sites/forbestechcouncil/2018/07/10/the-future-of-the-vpn-market`, July 2018.

[19] Gong, X., Borisov, N., Kiyavash, N., and Schear, N. Website detection using remote traffic analysis. In *Privacy Enhancing Technologies Symposium* (2012), Springer, pp. 58–78.

[20] Grover, S., and Feamster, N. The internet of unpatched things. *Proc. FTC PrivacyCon* (2016).

[21] Hoffman, P., and McManus, P. DNS Queries over HTTPS (DoH). RFC 8484, RFC Editor, October 2018.

[22] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and Hoffman, P. Specification for DNS over Transport Layer Security (TLS). RFC 7858, RFC Editor, May 2016.

[23] Inverse inc. Fingerbank. https://fingerbank.org/.

[24] Juarez, M., Imani, M., Perry, M., Diaz, C., and Wright, M. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security* (2016), Springer, pp. 27–46.

[25] Kastrenakes, J. Project Fi promises privacy with Google-run VPN. `https://www.theverge.com/2018/11/13/18089834/project-fi-enhanced-network-vpn-privacy-google-announcement`, November 2018.

[26] Kohno, T., Broido, A., and Claffy, K. C. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing 2*, 2 (2005), 93–108.

[27] Liu, J., Zhang, C., and Fang, Y. Epic: A differential

privacy framework to defend smart homes against internet traffic analysis. *IEEE Internet of Things Journal 5*, 2 (2018), 1206–1217.

[28] Mayer, J., Mutchler, P., and Mitchell, J. C. Evaluating the privacy properties of telephone metadata. *Proceedings of the National Academy of Sciences 113*, 20 (2016), 5536–5541.

[29] Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.-R., and Tarkoma, S. Iot sentinel: Automated device-type identification for security enforcement in iot. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on* (2017), IEEE, pp. 2177–2184.

[30] Murdoch, S. J., and Danezis, G. Low-cost traffic analysis of tor. In *Security and Privacy, 2005 IEEE Symposium on* (2005), IEEE, pp. 183–195.

[31] Nithyanand, R., Cai, X., and Johnson, R. Glove: A bespoke website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society* (2014), ACM, pp. 131–134.

[32] Park, H., Basaran, C., Park, T., and Son, S. H. Energy-efficient privacy protection for smart home environments using behavioral semantics. *Sensors 14*, 9 (2014), 16235–16257.

[33] Rist, O. The best VPN routers of 2018. https://www.pcmag.com/roundup/365023/the-best-vpn-routers, November 2018.

[34] Schmitt, P., Edmundson, A., and Feamster, N. Oblivious dns: Practical privacy for dns queries. *arXiv preprint arXiv:1806.00276* (2018).

[35] Shamsi, Z., Cline, D. B., and Loguinov, D. Faulds: A non-parametric iterative classifier for internet-wide os fingerprinting. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 971–982.

[36] Shamsi, Z., Nandwani, A., Leonard, D., and Loguinov, D. Hershel: single-packet os fingerprinting. In *ACM SIGMETRICS Performance Evaluation Review* (2014), vol. 42, ACM, pp. 195–206.

[37] Shmatikov, V., and Wang, M.-H. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security* (2006), Springer, pp. 18–33.

[38] Srinivasan, V., Stankovic, J., and Whitehouse, K. Protecting your daily in-home activity information from a wireless snooping attack. In *Proceedings of the 10th international conference on Ubiquitous computing* (2008), ACM, pp. 202–211.

[39] Stark, H. Your internet provider has already hacked your smart home. https://www.forbes.com/sites/haroldstark/2017/09/14/your-internet-provider-has-already-hacked-your-smart-home/, September 2017.

[40] Van Den Hooff, J., Lazar, D., Zaharia, M., and Zeldovich, N. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles* (2015), ACM, pp. 137–152.

[41] Verde, N. V., Ateniese, G., Gabrielli, E., Mancini, L. V., and Spognardi, A. No nat'd user left behind: Fingerprinting users behind nat from netflow records alone. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on* (2014), IEEE, pp. 218–227.

[42] Wang, T., Cai, X., Nithyanand, R., Johnson, R., and Goldberg, I. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security* (2014), pp. 143–157.

[43] Wang, T., and Goldberg, I. On realistically attacking tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies 2016*, 4 (2016), 21–36.

[44] Wang, T., and Goldberg, I. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium (USENIX Security 17)* (2017), pp. 1375–1390.

[45] Wang, W., Motani, M., and Srinivasan, V. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM conference on Computer and communications security* (2008), ACM, pp. 323–332.

Appendices

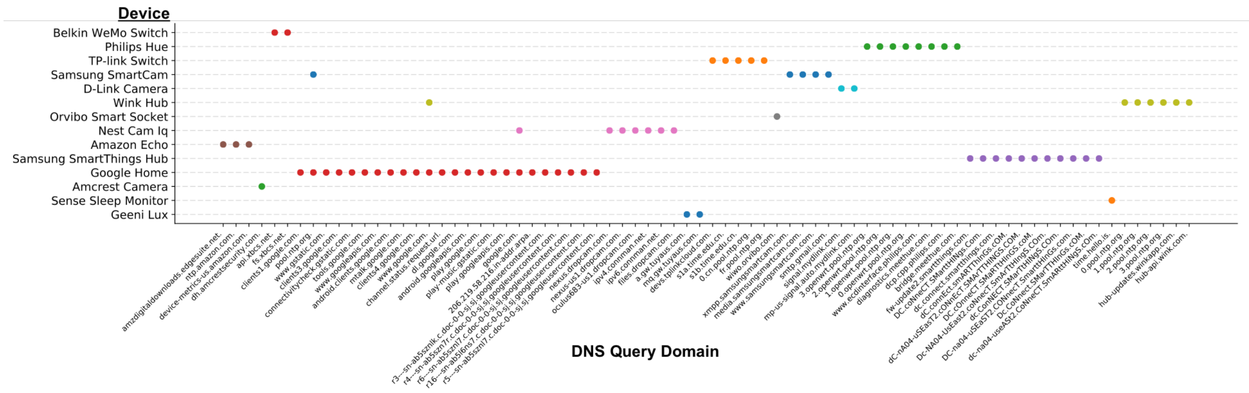

Fig. 6. All tested IoT devices send DNS requests for unique and mostly non-overlapping sets of domains. This allows the destination IP addresses of packets from these devices to serve as fingerprints for device identification.

IoT Device	Identifying DNS Query
Amcrest Security Camera	`dh.amcrestsecurity.com`
Amazon Echo	`device-metrics-us.amazon.com`
Belkin WeMo Switch	`prod1-fs-xbcs-net-1101221371`
D-Link Wi-Fi Camera	`signal.auto.mydlink.com`
Nest Cam Indoor	`nexus.dropcam.com`
Nest Cam IQ	`nexus.dropcam.com`
Orvibo Smart Socket	`wiwo.orvibo.com`
Phillips Hue Starter Set	`diagnostics.meethue.com`
Samsung SmartThings Hub	`dc.connect.smartthings.com`
Sense Sleep Monitor	`sense-in.hello.is`
TP-Link Smart Plug	`devs.tplinkcloud.com`
Wink Hub	`agent-v1-production.wink.com`

Table 2. DNS queries from smart home devices during a representative packet capture that are easily attributable to a specific device or manufacturer.

Device	Functionality	Description
Amazon Echo	limited	Can use as a bluetooth speaker with previously paired smartphone
		Echo recognizes "Alexa" keyword but does not provide any voice-control features
Belkin WeMo Switch	limited	Can turn switch on/off with physical button on device
		Cannot use smartphone app to control device even when phone on local network
Orvibo Smart Socket	limited	Can turn switch on/off with physical button on device or smartphone app on local network
TP-Link Smart Plug	limited	Can turn switch on/off with physical button on device or smartphone app on local network
Nest Security Camera	none	Unable to view video feed or receive detected motion notifications
Amcrest Security Camera	none	Unable to view video feed or control camera direction
Sense Sleep Monitor	none	Monitor does not record sleep data
		Light-based UI does not reflect local sensor readings
		Cannot use smartphone app to control device or access current data

Table 3. Tested commercially-available IoT devices had limited or no functionality when firewalled to prevent communication outside of the smart home LAN.

Device	Shaped upload traffic rate	Shaped download traffic rate	Avg. per-packet latency	Max. per-packet latency	Effects of traffic shaping
Amazon Echo	10 KB/s	10 KB/s	0.9s	4.4s	Echo answered questions with several seconds delay. With slower traffic, the Echo does not completely answer the question.
Belkin WeMo Switch	5.0 KB/s	2.5 KB/s	1.3s	2.6s	Works with few seconds of delay when switching from smartphone app. With slower traffic, the app loses connection to the switch.
Orvibo Smart Socket	0.25 KB/s	0.25 KB/s	0.8s	2.6s	Works with few seconds of delay when switching from smartphone app. With slower traffic, the app registers switching success, but the socket does not actually change state.
TP-Link Smart Plug	0.5 KB/s	0.5 KB/s	0.8s	1.7s	Works with few seconds of delay when switching from smartphone app. With slower traffic, the app reported that the plug was unreachable.
Nest Security Camera	10 KB/s	10 KB/s	8.5s	17.6s	Video streamed with 10-15 seconds of lag and app reported slow upload speeds. With slower traffic the app lost connection with the camera.
Amcrest Security Camera	35 KB/s	2.5 KB/s	1.5s	3.4s	Video streamed with 1 minute delay, but no breakage. Could tolerate slower traffic, but with $>$ 3 minute delay and less reliability.
Sense Sleep Monitor	2.5 KB/s	2.5 KB/s	0.4s	1.4s	The smartphone app could still receive live updates. With slower traffic, live updates stop.

Table 4. Lowest tested shaped traffic rates and corresponding per-packet latencies at which smart home devices still function, dependent on usage patterns and traffic load on the shaper. Reported per-packet latencies were measured in the upload direction. Most tested device were more latency-tolerant in the download direction.

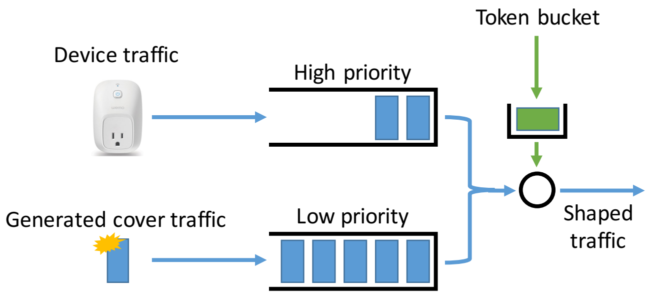

Fig. 7. Illustration of our traffic shaping implementation for periods of constant-rate padding during STP, including cover traffic generation and traffic control in the kernel. Device packets in the high priority queue are always sent before cover packets in the low priority queue. Cover packets are generated faster than the shaped rate, ensuring that cover packets are always present in the low priority queue. The token bucket shaper has a buffer size of 1 token. The rate of token arrival is set to the shaped traffic rate.