MAC-Layer Traffic Shaping Defense Against WiFi Device Fingerprinting Attacks

Mnassar Alyami,[1,2] Mohammed Alkhowaiter,[1] Mansour Al Ghanim,[1] Cliff Zou,[1] and Yan Solihin[1]

[1]*College of Engineering and Computer Science, University of Central Florida, USA*
{mnassar.alyami@knights. | mok11@knights. | malghanim@knights. | changchun.zou@ | yan.solihin@}ucf.edu
[2]*College of Computer Science and Information Technology, Jazan University, Saudi Arabia*
malsaad@jazanu.edu.sa

***Abstract*—WiFi networks are vulnerable to statistical traffic analysis attacks, even when a WiFi network is securely encrypted and the attacker is unable to join the network. Many defenses proposed in the literature are inefficient to deal with profiling attacks against WiFi-based Internet-of-Things (IoT) devices, because they burden the Internet traffic with high bandwidth overhead and pose deliberate delay on packet transmission. In this paper, we propose a new MAC-layer packet injection technique where injected dummy packets only exist within the WiFi link between IoT devices and their connected WiFi access point. This traffic shaping defense is effective against data-link device profiling attacks without adding any Internet-side overhead or time delay in legitimate traffic. We evaluated our approach on four WiFi-based IoT devices against a recent privacy attack, and showed the reduction of attack classification accuracy from the original 100% to 54%, close to random guessing.**

***Index Terms*—Device Fingerprinting, IoT Privacy; Traffic Analysis Countermeasure**

I. Introduction

IEEE 802.11 Wireless network (WiFi) is widely adopted to support Internet-of-Things (IoT) environments; the emerging paradigm has gained rapid deployment but has increased privacy concerns. Device Fingerprinting (DF) attacks allow observers to identify IoT devices and their operation status in a wireless network, which implies a severe privacy threat due to their specialized tasks. For instance, an eavesdropper can monitor events of the thermostat transitions (i.e., Home to Auto-Away and vice versa) from network traffic to infer when householders are entering/leaving their residential buildings [1]. Indeed, the DF attack is a well-established privacy threat in the literature from monitoring network traffic at the network-layer [2]–[4] and the data-link layer [5], [6].

Source anonymity technologies [7], [8] can be used to ensure the unlikability of the traffic to the destination to avoid *network-layer* traffic analysis by a remote attacker. Yet, *link-layer* traffic analysis attacks (i.e., local eavesdropping) are still undefeated. The attackers who have proximity to the access point (AP) can, without joining the network, collect WiFi traffic and extract statistical characteristics (e.g., frame sizes, timing) to match observed patterns to known patterns of IoT devices. Inference attacks from WiFi eavesdropping are not as weak as one can imagine; this attack is very realistic and can be performed by practically anyone with close proximity to the WiFi network [6].

There has been active research work to mitigate the privacy leakage from network traffic analysis. (We will discuss this more in Section II.) Broadly speaking, these defensive systems fall into one of three categories: Uniform, Randomization, and Mimicry-based obfuscations. *Uniform-based* methods enforce a fixed packet rate [9], whereas *randomization* techniques randomize packet size distribution and inter-arrival time to make network traffic patterns odd and ”unmatchable” to the learned signatures [10]. *Mimicry-based* approaches let the system copy a target pattern to force a model used for packet classification to misclassify [11]. Nevertheless, none of these strategies provide an efficient way to remove (or hide) statistical fingerprints without posing a delay on packets. Although some IoT applications are delay-tolerant, other time-sensitive applications would be significantly affected (e.g., alarming systems) by such defenses. Hence, it is extremely important to provide a general privacy-preserving solution to accommodate all types of IoT devices.

Furthermore, with billions of connected “things” over the Internet providing services to governments, hospitals, businesses, and individuals, Internet service providers (ISP) in some areas may limit the bandwidth to ease congestion over the network. With that in mind, relying on a mechanism that increases the *internet traffic* [12], [13] would deplete the network resources and harm user experience. Therefore, existing countermeasures that increase internet bandwidth are not practical for wide deployment.

Towards this end, this paper presents a new MAC-layer traffic shaping defense against device fingerprinting attacks with the following properties:

- Effective: We present a confusion-based obfuscation approach that defeats device fingerprinting attacks by making a pair of devices look indistinguishable. Furthermore, our solution is simple to implement on IoT devices and AP.
- Zero internet bandwidth overheads: Our approach brings the obfuscation technique to the WiFi network and contains injected dummy packets within the local WiFi network, preventing an increase in internet traffic bandwidth. We accomplish this by enabling AP and IoT devices to recognize and drop the dummy traffic generated by our defense, hence dummy traffic does not cause internet

bandwidth overhead. In addition, the injector creates and transmits dummy packets without tampering with the original traffic of a device, resulting in little time delay.

The remainder of this paper is organized as follows: We review related work in Section II. In Section III, we present the threat model and outline a recent DF attack. Our traffic obfuscation methodology is described in Section IV. We evaluate our technique and discuss our results in Section V and Section VI, respectively. Finally, we draw our conclusion and discuss future works in Section VII.

II. Related Work

Privacy leakage from analyzing encrypted traffic has been extensively studied, including identification (or fingerprinting) Apps [14]–[16], websites [17]–[19], and devices [2]–[6]. An increasing number of studies have found that monitoring IoT devices can yield surprisingly detailed information about user activity. Recent work [20] has shown that monitoring a camera's video traffic would allow network observers to infer changes in user activity (e.g., moving from sitting to walking).

Several papers introduced countermeasures against various privacy attacks such as internet traffic tracing [7], app identification [21], and website fingerprinting [12]. Are these countermeasures effective against data-link device fingerprinting (DF) attacks? We center our answer on two aspects: the effectiveness of mitigating the privacy leakage from statistical traffic analysis and the feasibility of implementing the defense in terms of time and bandwidth overhead. That leads us to argue that all the prior defenses are ineffective against our attack as some defenses do not extend to our threat model [7], [21] (i.e., they are tailored towards a different scope), others are expensive to implement [12]. We discuss below related work and their limitations to our attack scenario.

Effective defenses, though with high overheads (i.e., >150% bandwidth and/or latency), were proposed in response to newer website fingerprinting attacks [9], [22], [23]. Later, Wang et al. [12] developed a practical solution that is well-appreciated by The Onion Router (Tor) community to mitigate the privacy leakage at lower overheads. They modified the proxy and client's browser to maintain a buffer that holds burst sequences to be molded with each other to form a non-sensitive pattern at little bandwidth instead of injecting additional dummy packets per burst. Still, their approach requires over 30% overhead in both bandwidth and time. Additionally, Tor uses fixed-length packets to deliver information, unlike IoT traffic that varies in length. Observers can perform the attack by solely leveraging packet length statistics, as reported by [24]. Therefore, efficient countermeasures designed for Tor cannot be extended to IoT networks. However, other solutions that distort only packet length features [21] are ineffective either; packet interarrival time features are sufficient to fingerprint the types of devices using deep learning algorithms [4]. Our obfuscation technique, on the other hand, obscures all statistical features in the traffic and does not increase the internet traffic.

As mentioned by Xiong et al. (2022) [25], IoT networks require special tailoring as current countermeasures have tended to focus on shaping web-level traffic rather than device-level. Unlike web servers, IoT devices are resource-constrained. This led the authors in [25] to design a tunable privacy model for IoT network traffic so users can balance the trade-off between their privacy demands and resource requirements. Nevertheless, their defense is limited to obfuscating event-indicating traffic (e.g., a sensor detects motion), so attackers can still fingerprint IoT devices. In this work, we protect against DF attacks, which implies the immunity against the event-level adversary.

Our literature search was able to identify a single paper [5] in which the authors attempted to defend against the data-link profiling attack. The authors of [5] evaluated a traffic injection solution and reported a reduction in the accuracy from 94% to 15%. The main limitation of their work is that they rely on the assumption that the attacker does not know about the defense, which is an unreasonable assumption that can lead to significant privacy leakage. Attackers can implement their technique and adversarially train a better classifier to overcome the defense. Different from their work, we applied our technique to both training and testing data to show the more robust technique against future attacks.

III. Preliminaries

In this section, we discuss the threat model we consider. We then give a brief description of our recent privacy attack [6].

A. Threat Model

In this paper, we are concerned with the privacy leakage of IoT devices from statistical traffic analysis attacks. As shown in Fig. 1, we assume an observer positioned within the signal range of the victim's AP to record encrypted WiFi traffic. This adversary does not need to join or break into the WiFi network to perform the attack; thus, the observer can only access the encrypted data-link traffic (i.e., MAC addresses, signal strength, observation timestamp, frame length, and type).

Our adversary's goal is to infer the device type (e.g., Baby monitor, Security camera, etc.). We consider a closed-world scenario, where the attacker is interested in profiling a particular set of devices to infer sensitive information or monitor user activity. For example, the attacker can infer whether a unit has expensive equipment in it or not, and roughly how many people being active in that unit. IoT monitoring can also reveal user activity. If the attacker succeeds in fingerprinting

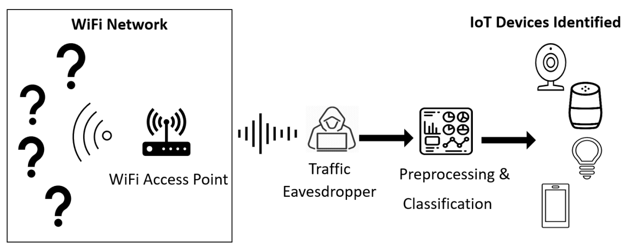


Fig. 1. Threat Model.

sleep monitors, it is then possible to extend the traffic monitoring to observe an event packet stream from packet lengths, triggered when the user is sleeping. This assumption has been successfully validated recently [25].

We assume the observer is aware of the defense, thus can adversarially train his/her classifier to overcome our defense. To counter against further attacks successfully, we reduce the attack success near to random guessing.

B. WiFi-based Device Fingerprinting Attack

Our prior work presented a practical privacy attack that allows eavesdroppers to identify IoT devices in a WiFi network with high accuracy (95+%) [6]. The adversary exploits the statistical differences in IoT devices' network traffic due to their various and distinct functionalities (e.g., streaming, sensing). From the machine learning perspective, device identification is a classification problem in which the adversary captures raw packets of his/her own devices and then extracts distinguishable features to build a profile that can be used for identifying IoT devices in-the-wild with similar data patterns.

In this attack, we show the attacker can set a commodity laptop in monitor mode to passively record encrypted IoT network traffic over a short time (30 seconds). The attacker then pre-processes the trace by filtering out all control and management MAC-layer frames to exploit data patterns from the remaining data type Mac-layer frames.

Next, we obtain the pre-processed data to extract 14 statistical features f from encrypted WiFi traffic for profiling:

- f_1: Number of incoming frames from AP to the device.
- f_2: Number of outgoing frames from the device to AP.
- f_3: The variance of inter-arrival time of the received frames from AP.
- f_4: The average number of frames the device transmitted consecutively prior to receiving a data frame.
- f_5: The average number of frames the device received consecutively prior to transmitting a data frame.
- f_6: Flow size in bytes of transmitted data frames from the device.
- f_7: Flow size in bytes of received data frames from the device.
- f_8: The number of distinct lengths in device transmitted data frames.
- f_9: The number of distinct lengths in device received data frames.
- f_{10}: Maximum observed frame length.
- f_{11}: The most frequent length in transmitted data frames from the device (i.e., the length that appears most in the monitoring window).
- f_{12}: The most frequent length in received data frames from AP.
- f_{13}: The variance of device transmitted data frames.
- f_{14}: The variance of device received data frames.

Finally, we use the summary data approach we summarized in Section V to create a dataset of 14 columns; each represents a feature from f. These statistical data are found to be effective signatures to feed into machine learning (ML) classification to infer information about the device type and its working activity (busy vs. idle). In [6], we evaluated 3 ML algorithms: Support Vector Machine, Naïve Bayes, and Random Forest. The latter model (i.e., Random Forest) outperforms the other two algorithms and achieves an average F1 score of 95%.

IV. Obfuscation via Confusion

Our proposed traffic shaping defense is based on dummy packet injection to make two different devices look very similar. It sends dummy packets in a way that masquerades another device and blends into the original traffic so that the attacker cannot distinguish which of the two devices the observed traffic belongs to. To be specific, let a classifier C that classifies a device D using its traffic T_D. Our defense shapes T_D to $T\prime_D$ to confuse C such that it cannot classify D correctly.

To achieve this, we shape the original traffic of a device x, $T_{D(x)}$, by injecting a *traffic pattern* from another device y, $\mathcal{P}_{D(y)}$, so that we obtain the shaped traffic as:

$$T\prime_{D(x)} = T_{D(x)} + \mathcal{P}_{D(y)}, \tag{1}$$

where $\mathcal{P}_{D(y)}$ is a segment of trace recorded from another device y. Likewise, for the device y and its original traffic $T_{D(y)}$, we use device x's traffic pattern to shape it, which means the shaped traffic of device y is:

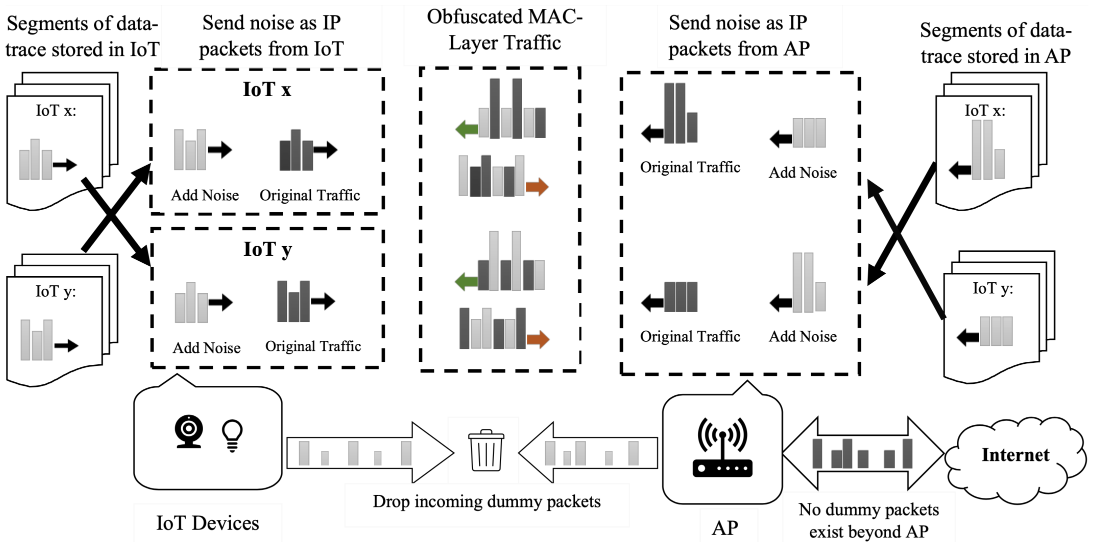
$$T\prime_{D(y)} = T_{D(y)} + \mathcal{P}_{D(x)}. \tag{2}$$

As a result, classifier C cannot distinguish device x from device y because $T\prime_{D(x)} \equiv T\prime_{D(y)}$. This is empirically true as the pattern in a recorded trace $\mathcal{P}_D$ is equivalent to the pattern in the online traffic observed in T_D.

Furthermore, we assume the adversary knows about our defense being performed at the WiFi network; therefore, if we adversarially train $C\prime$ using $T\prime_D$, then $C\prime$ also cannot classify D as $T\prime_{D(x)} \equiv T\prime_{D(y)}$.

In our wireless communication scenario, two objects are involved in packet injection (i.e., AP and a client device). We adopt a one-way injection at each node to obfuscate the bi-directional traffic between the AP and client. Thus, we have a two-way injection in the entire network. In other words, the AP sends noisy traffic to the client, and the client sends noisy traffic to the AP. As a result, the incoming and outgoing traffic is thoroughly obfuscated.

Fig. 2 depicts the traffic shaping scenario between the AP and two IoT devices. The system enables the AP and IoT to store packet traces to mask the traffic. On the right hand, the AP injects dummy packets into IoT(x) traffic with sending patterns derived from data traces recorded from IoT(y). The same is performed for the second device (IoT(y)) but using the corresponding packet trace (IoT(x)). On the left hand, each device also uses stored packet traces to mimic the behavior of each other when responding to the AP. All noisy traffic is sent at the network-layer. Hence, all packets are encrypted at the data-link layer. Therefore, the WiFi traffic is fully obfuscated, and the wireless eavesdropper cannot identify and filter out those dummy packets. Finally, all generated dummy



Segments of data-trace stored in IoT
Send noise as IP packets from IoT
Obfuscated MAC-Layer Traffic
Send noise as IP packets from AP
Segments of data-trace stored in AP
IoT x:
IoT x
Add Noise
Original Traffic
Original Traffic
Add Noise
IoT y:
IoT y
Add Noise
Original Traffic
Original Traffic
Add Noise
Drop incoming dummy packets
Internet
IoT Devices
AP
No dummy packets exist beyond AP

Fig. 2. WiFi traffic shaping scenario between AP and two IoT devices.

flow is dropped by the AP and IoT devices when they receive them, which means the dummy traffic only exists in the WiFi network, never appears beyond the AP to the Internet. In this way, our proposed approach does not add any bandwidth overhead on the Internet side.

A. Algorithm

A pseudo-code for the main loop of our traffic shaping defense is shown in Algorithm 1. Both the AP and client run the same loop but using the corresponding data trace (i.e., incoming or outgoing). As mentioned before, the data trace is recorded offline from a device to provide the basis for the mimicry process.

The input of data trace is stored in a sequence $Seq\{\}$, which contains a set of segments S. Each individual segment holds a series of packets $\{Pkt_0^n\}$ captured over a specific time window.

The program, once executed, feeds the injector with a segment of packets S_x. In this manner, the injector uses the two values provided from each packet in the segment (i.e., timestamp and length) to decide each packet's length and insertion schedule (i.e., waiting time).

We randomize S_x using two operations. First, we add a marginal random value $\in \{0.0$ to $0.09\}$ to each time interval. Second, we run a pseudorandom number generator to get two lists L_{Drop} and $L_{Replace}$ of the same size as S_x. Both list elements $\in [0, 1]$ with a random probability $Prob$ from 0% to 5% to be ones. So, when injecting the dummy packets from S_x, we skip the i^{th} element of S_x when the i^{th} element in $L_{Drop} = 1$. However, if the corresponding element in $L_{Drop} = 0$ and the i^{th} element in $L_{Replace} = 1$, the program will replace the packet from S_x with a random size packet that is less than or equal to the maximum transmission unit (MTU) and send it after a random waiting time, up to 0.09 seconds. Otherwise (i.e., when both elements in L_{Drop} and $L_{Replace} = 0$), the program will inject form S_x, which will be at least 95% of the time. Thus, we slightly randomize the time-based and flow-based features of packet sequences; hence every time we use the same segment, it looks different. This randomization is necessary to prevent the attacker from conducting frequency analysis to identify dummy packets (i.e., using the same segment at a fixed pattern).

Algorithm 1 Mimicry-Based Traffic Injection

```
 1: $Seq\{\} \leftarrow$ multiple segments $S$
 2: while True do
 3:     $DummyPkt \leftarrow$ Source, Destination, flag = 'Dummy'
 4:     $S_x \leftarrow$ Randomly select a segment from $Seq$
 5:     $Volume \leftarrow$ length of $S_x$
 6:     $Prob \leftarrow$ random (low = 0, high = 5)
 7:     $L_{Drop}$ []$\leftarrow$ random (value $\in$ {0, 1}, size = $Volume$,
        probability = ($Prob$, value = 1) ).
 8:     $L_{Replace}$ []$\leftarrow$ random (value $\in$ {0, 1}, size = $Volume$,
        probability = ($Prob$, value = 1) ).
 9:     for $i \leftarrow 1$ to $Volume$ do
10:         if $L_{Drop}[i] = 1$ then
11:             continue
12:         else if $L_{Replace}[i] = 1$ then
13:             Waiting time = random (low = 0, high = 0.09)
14:             send $DummyPkt$ (size = random)
15:         else
16:             $Interval \leftarrow S_x : Pkt_i - Pkt_{i-1}$
17:             $r$ = random (low = 0, high = 0.09)
18:             Waiting time = $Interval + r$
19:             send $DummyPkt$ (size = $S_x : Pkt_i$)
20:         end if
```

B. Implementation

This section discusses practical considerations involved in implementing our defense and noise generation prototype.

a) Practical Considerations: In creating the dummy packets, we need to design a mechanism that allows IoT devices and the AP to easily identify and discard the noisy frames. To achieve this, we propose putting a flag on the header of dummy packets. We exploit the reserved/unused bit flag on the IP header to label dummy packets. Since the adversary observes only the data-link layer traffic in an encrypted WiFi network, the IP header is fully encrypted, and hence, the attacker cannot observe the IP header to filter out dummy packets. The tagged packets can be ignored in different ways, such as using the *ip* command *blackhole* in Linux OS[1], which is used to drop packets silently.

b) Noise Generation Prototype: Since we cannot modify IoT devices due to firmware restrictions, we develop a proof-of-concept implementation on two Linux computers; the first is on the AP mode to emulate as an AP and the second one emulates as an IoT device. We developed a Python script with the *scapy* library[2] to generate and send dummy packets between the two computers over a WiFi connection. The packet creation and transmission are implemented using the $IP()$ and $send()$ functions at layer 3. Our source code can be found on GitHub[3].

V. Evaluation

We evaluate the performance of our obfuscation technique by simulating the noise injection on a data trace collected over 30 minutes from four WiFi-based IoT devices (Baby monitor, Camera, Light bulb and Smart plug). Note we capture additional data trace from the same devices to be used as background noise. We opted for a 30-minutes sample size due to the observed persistent pattern by all devices, which is also noted in [6]. Fig. 3 gives two examples of these steady patterns from two devices (Light bulb and Smart plug). The spikes indicate packet arrivals at different times. The larger the spike, the more packets observed during a 1-sec time window. We can notice that both devices exhibit a consistent pattern (regular number of packets per second), with higher traffic volume when the device is turned ON/OFF. For example, the light bulb on the left char mostly has a frequent number of packets/sec of 2.5 and increases above 5 when changing its working status. Besides, the time difference between packet streams is remarkably constant. Therefore, we believe that our data has a sufficient mix of testing signatures to prove the concept of our defense.

To construct our dataset, we adopted the summary-data format described in [6]; For every device (i.e., every unique MAC address)[4], we calculate 14 features $\{f_1, f_2, \ldots, f_{14}\}$, as outlined in Section III-B, observed over a 30-seconds time window. For example, to obtain the first datapoint, we calculate f during the time window $W_1 = \{0, \ldots, 30\}$. Then, we slide the time window forward by 30 seconds to obtain the second datapoint within $W_2 = \{30, \ldots, 60\}$ and repeat the process until the end of the trace. (We refer readers to more detailed descriptions in [6]). Note that we created two datasets: the first represents the undefended traffic, and the second is constructed after shaping the traffic using our proposed technique.

For evaluation, we randomly split our datasets into 75% for training and 25% for testing. We evaluate our defense against the best classifier (using Random Forest) reported in [6]. Below, we discuss our evaluation metrics and compare our results with random guessing as a baseline.

A. Evaluation Metrics

We evaluate our defense using the same standard metrics used for evaluating our prior attack: Accuracy, Precision, Recall and F1 Score. The detailed calculation of these evaluation metrics is given in the Appendix.

Our goal is to confuse the classifier from predicting the true device. Hence, we show the indistinguishability of the four devices under investigation through a lower score in all metrics compared to our attack, ideally close to random guessing[5] (50%).

Additionally, we evaluate the overhead of our defense by calculating the percentage of added dummy data, which we define as:

$$Overhead = \frac{D}{R} \tag{3}$$

where D is the total amount of dummy packets, and R is the total amount of real packets.

Generally, inserting extra dummy packets affects the internet bandwidth, but as we mentioned earlier, our technique avoids this limitation by dropping noise packets at the AP before entering the Internet. However, we define this metric to simplify the analysis of the extra burden laid on the WiFi network or the devices (e.g., power consumption overhead).

B. Results

Table. I compares the accuracy, precision, recall and F1 score of two classifiers: the first is trained using the original traffic, which represents the attack performance and the second is trained using the defended/shaped traffic (i.e., After injecting dummy packets). We can observe the reduction of all metrics from 100% to 54% on average by our defense.

Furthermore, Table. II shows how the classifier confuses and misclassifies one device for another in our testbed (Bulb vs. Plug and Baby Monitor vs. Camera). The model could not learn from the "mix" as we inject from the bulb to the plug and vice versa. Likewise, we pair the traffic of the baby monitor and camera. As a result, a potential attack classifier will have difficulty learning from data created in this fashion.

The same Table (Table. II) also shows the percentage of dummy packets to the real packets as 106%. This is expected as the WiFi network traffic doubles when we program each device to duplicate another device's traffic.

[1]https://man.archlinux.org/man/ip-rule.8

[2]https://pypi.org/project/scapy/

[3]https://github.com/MnassarAlyami/Mimicry-Based-Traffic-Injection.git

[4]The basic assumption here is that a device will not change its MAC address during this observing time window, which is true for current IoT devices.

[5]Random guessing achieves $1/k$ accuracy where k is the number of classes in the model. Since we confuse the attacker between two devices at a time, then $k = 2$.

TABLE I
CLASSIFICATION ACCURACY, PRECISION, RECALL AND F1 SCORE OF TWO DATASETS.

Device	No Defense (%)				Defended Traffic (%)			
	Accuracy	*Precision*	*Recall*	*F1*	*Accuracy*	*Precision*	*Recall*	*F1*
Baby Monitor	100	100	100	100	60	56	60	58
Bulb	100	100	100	100	50	50	50	50
Camera	100	100	100	100	53	57	53	55
Plug	100	100	100	100	50	50	50	50

TABLE II
CONFUSION MATRIX OF OUR IOT DEVICES CLASSIFICATION AGAINST OUR DEFENSE AND OVERHEAD.

		Actual			
		Baby Mon.	Bulb	Camera	Plug
Prediction	Baby Mon.	**0.6**	0.0	0.47	0.0
	Bulb	0.0	**0.5**	0.0	0.5
	Camera	0.4	0.0	**0.53**	0.0
	Plug	0.0	0.5	0.0	**0.5**
Balanced Accuracy		54%			
Added Packets		106%			

VI. DISCUSSION

Privacy leakage mitigation: Our results validate that our traffic-confusion-based obfuscation technique can defeat data-link device fingerprinting attacks. We show the four devices in our experiments become indistinguishable for the classifier, although it was trained using the obfuscated traffic. The learned model resulted in 54% accuracy compared to 50% using random guessing. This confusion is expected as signatures in both training and testing data are very similar for each device.

The side-channel leakage from encrypted traffic analysis is not only a privacy leakage but also a critical information leakage to vulnerability scanners. Mapping the high fidelity classification information to known vulnerable devices [26] provides a potential attack plan for when the adversary wishes to penetrate the environment. This preparation phase can provide insight to the adversary on what tools and penetration assets to establish a foothold in the target environment. Hence, this defense may also provide *security* protection as privacy and security threats are sometimes related.

Overhead: Our method provides privacy without incurring internet bandwidth overhead. Moreover, we rely on pre-recorded traces to generate packet sequences to enforce confusion against inference attacks, which carries a small computational cost. However, since our defense requires additional packets to obfuscate traffic patterns, that would introduce an inevitable power consumption overhead.

We propose a potential obfuscation mechanism without inserting dummy packets to overcome the power consumption overhead. Shaping the IoT traffic via virtual identifiers [27] would allow each IoT device to generate multiple virtual network interface cards; each has a different MAC address to communicate with AP. As a result, the network traffic of one device will be distributed among different identifiers. Thus, it becomes much more difficult for the attacker to link packets from different identifiers for profiling. However, there are still unresolved practical challenges with this privacy model. For example, how to decompose the internet traffic of a device (e.g., HTTP requests) into fractions over multiple virtual interfaces with different IP addresses? We plan to conduct further investigation in this direction as future work.

VII. CONCLUSION AND FUTURE WORK

The existing countermeasures against network traffic analysis attacks at the network-layer are not a viable option against data-link DF attacks. One reason is that they would negatively impact the internet condition with significant noisy traffic. Besides, prior studies present lightweight solutions in bandwidth but at a higher latency. By adding noise traffic only on the WiFi link, this paper presents a zero-overhead defense in internet bandwidth against WiFi-based traffic analysis attacks. In addition, our technique injects dummy packets without posing a deliberate delay on the original traffic of a device. We present a traffic obfuscation mechanism in which a pair of devices become indistinguishable. Each device sends dummy packets to mimic the other, so it generates two patterns simultaneously to hamper the fingerprinting attack. Our method significantly reduces the classification accuracy of a recent privacy attack to be close to random guessing.

For future work, we aim to answer the question of how much privacy is enough? In other words, we plan to introduce a risk assessment model to allow users to balance between the cost of implementing the countermeasure (i.e., power consumption) and privacy. Since different IoT devices may leak sensitive information that varies in severity, such as the user presence/absence, age, health condition, etc., it is therefore of great interest to quantify privacy leakage with respect to the device type to allow for an optimum balance between privacy protection and overhead.

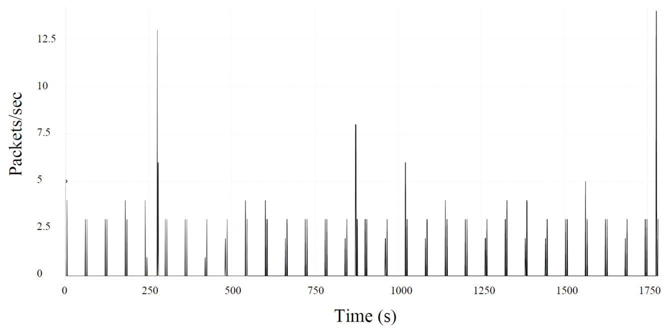
ACKNOWLEDGMENT

This work was sponsored by the U.S. National Science Foundation (NSF) under Grant DGE-1915780. The authors also would like to thank Mr. Karl Ackerman from Sophos Inc. for the helpful discussion.

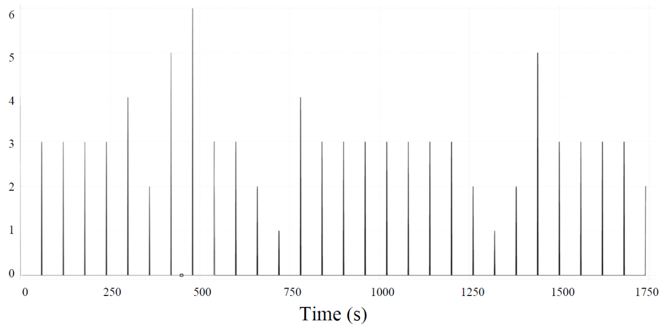
REFERENCES

[1] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. Is anybody home? inferring activity from smart home network traffic. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 245–251. IEEE, 2016.

[2] 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*, 18(8):1745–1759, 2018.



(a) Smart light bulb.



(b) Smart plug.

Fig. 3. Traffic flow of two devices over 30 minutes.

[3] Mustafizur R Shahid, Gregory Blanc, Zonghua Zhang, and Hervé Debar. Iot devices recognition through network traffic analysis. In *2018 IEEE international conference on big data (big data)*, pages 5187–5192. IEEE, 2018.

[4] Sandhya Aneja, Nagender Aneja, and Md Shohidul Islam. Iot device fingerprint using deep learning. In *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pages 174–179. IEEE, 2018.

[5] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 207–218, 2020.

[6] Mnassar Alyami, Ibrahim Alharbi, Cliff Zou, Yan Solihin, and Karl Ackerman. Wifi-based iot devices profiling attack based on eavesdropping of encrypted wifi traffic. In *2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*, pages 385–392, 2022.

[7] 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.

[8] Asma Iman Kouachi, Abdelmalik Bachir, and Noureddine Lasla. Anonymizing communication flow identifiers in the internet of things. *Computers & Electrical Engineering*, 91:107063, 2021.

[9] Xiang Cai, Rishab Nithyanand, and Rob Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 121–130, 2014.

[10] Brandon Wiley. Dust: A blocking-resistant internet transport protocol. *Technical rep ort. http://blanu. net/Dust. pdf*, 2011.

[11] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9. Citeseer, 2009.

[12] Tao Wang and Ian Goldberg. {Walkie-Talkie}: An efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1375–1390, 2017.

[13] Ahmed Abusnaina, Rhongho Jang, Aminollah Khormali, DaeHun Nyang, and David Mohaisen. Dfd: adversarial learning-based approach to defend against website fingerprinting. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2459–2468. IEEE, 2020.

[14] John S Atkinson. *Your WiFi is leaking: inferring private user information despite encryption.* PhD thesis, UCL (University College London), 2015.

[15] Qiuning Ren, Chao Yang, and Jianfeng Ma. App identification based on encrypted multi-smartphone sources traffic fingerprints. *Computer Networks*, 201:108590, 2021.

[16] Liuqun Zhai, Zhuang Qiao, Zhongfang Wang, and Dong Wei. Identify what you are doing: Smartphone apps fingerprinting on cellular network traffic. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2021.

[17] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1187–1203, 2016.

[18] Chenggang Wang, Jimmy Dani, Xiang Li, Xiaodong Jia, and Boyang Wang. Adaptive fingerprinting: website fingerprinting over few encrypted traffic. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 149–160, 2021.

[19] Meng Shen, Zhenbo Gao, Liehuang Zhu, and Ke Xu. Efficient fine-grained website fingerprinting via encrypted traffic analysis with deep learning. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10, 2021.

[20] Jinyang Li, Zhenyu Li, Gareth Tyson, and Gaogang Xie. Your privilege gives your privacy away: An analysis of a home security camera service. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 387–396. IEEE, 2020.

[21] Louma Chaddad, Ali Chehab, Imad H Elhajj, and Ayman Kayssi. Optimal packet camouflage against traffic analysis. *ACM Transactions on Privacy and Security (TOPS)*, 24(3):1–23, 2021.

[22] 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.

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

[24] Antônio J Pinheiro, Jeandro de M Bezerra, Caio AP Burgardt, and Divanilson R Campelo. Identifying iot devices and events based on packet length from encrypted traffic. *Computer Communications*, 144:8–17, 2019.

[25] Sijie Xiong, Anand D. Sarwate, and Narayan B. Mandayam. Network traffic shaping for enhancing privacy in iot systems. *IEEE/ACM Transactions on Networking*, pages 1–16, 2022.

[26] Yinxin Wan, Kuai Xu, Guoliang Xue, and Feng Wang. Iotargos: A multi-layer security monitoring system for internet-of-things in smart homes. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 874–883. IEEE, 2020.

[27] Omar Nakhila, Muhammad Faisal Amjad, Erich Dondyk, and Cliff Zou. Gateway independent user-side wi-fi evil twin attack detection using virtual wireless clients. *Computers & Security*, 74:41–54, 2018.

Appendix

To evaluate our defense, we used four metrics: Accuracy, Precision, Recall and F1 Score, which can be calculated as follows:

$$Accuracy = \frac{T}{T+F} \tag{4}$$

where T is the number of correct predictions, and F denotes false predictions.

$$Precision = \frac{TP}{TP+FP} \quad (5) \qquad Recall = \frac{TP}{TP+FN} \quad (6)$$

where TP means true positives, TN is true negatives, FP is false positives, and FN is false negatives.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$