Attacking and Protecting Tunneled Traffic of Smart Home Devices

Ahmed Alshehri
Colroado School of Mines
Golden, Colorado
alshehri@mines.edu

Jacob Granley
Colroado School of Mines
Golden, Colorado
jgranley@mines.edu

Chuan Yue
Colroado School of Mines
Golden, Colorado
chuanyue@mines.edu

ABSTRACT

The number of smart home IoT (Internet of Things) devices has been growing fast in recent years. Along with the great benefits brought by smart home devices, new threats have appeared. One major threat to smart home users is the compromise of their privacy by traffic analysis (TA) attacks. Researchers have shown that TA attacks can be performed successfully on either plain or encrypted traffic to identify smart home devices and infer user activities. Tunneling traffic is a very strong countermeasure to existing TA attacks. However, in this work, we design a Signature based Tunneled Traffic Analysis (STTA) attack that can be effective even on tunneled traffic. Using a popular smart home traffic dataset, we demonstrate that our attack can achieve an 83% accuracy on identifying 14 smart home devices. We further design a simple defense mechanism based on adding uniform random noise to effectively protect against our TA attack without introducing too much overhead. We prove that our defense mechanism achieves approximate differential privacy.

KEYWORDS

Internet of Things (IoT); Traffic Analysis (TA); Privacy; Differential Privacy; Smart Homes; Attacks; Defenses

ACM Reference Format:
Ahmed Alshehri, Jacob Granley, and Chuan Yue. 2020. Attacking and Protecting Tunneled Traffic of Smart Home Devices. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy (CODASPY '20), March 16–18, 2020, New Orleans, LA, USA.* , 12 pages. https://doi.org/10.1145/3374664.3375723

1 INTRODUCTION

The adoption of IoT (Internet of Things) devices for personal use has been growing dramatically. According to a McKinsey report on smart homes, there were 29 million smart homes in the U.S. in 2017, 31% more than the previous year [36]. This growth is continuing as many companies, such as Amazon, have recently launched a multitude of new smart home devices and appliances [38]. However, owners of smart homes are still not fully aware of what is at stake when it comes to personal privacy.

Many users might not know how their personal data is collected and shared, as this is often done behind the scene. New regulations, such as the European General Data Protection Regulation (GDPR), could help users understand how their data is dealt with and give some options to opt out of data collection or modify incorrect information [40]. These regulations help defend against attacks or misuses of data that are detectable, but may not be effective against traffic analysis (TA) attacks, which do not alter any data. However, TA attacks have been shown to be very effective in compromising user privacy in both Web and IoT environments (Section 2).

In the IoT environments, one key objective of TA attacks is to identify smart devices. This is because successful device identification is the first and the foremost step in both the inference of user activities for compromising privacy, and the intrusion of devices for compromising security.

Existing TA attacks in the IoT environments are largely flow-based as they build network traffic flows by exploiting metadata such as the source and destination addresses of packets, time intervals of packets, and network protocols to further identify the devices [1, 3, 28]. A common defense approach against flow-based TA attacks is traffic tunneling, which hides the metadata and masks the flow information. Over the tunneled traffic, only three features, i.e. packet size, direction, and time, are accessible to attackers. While this makes TA attacks much harder to be successful, tunneling alone is not enough for protecting user privacy. Apthorpe et al. suggested that TA attacks may still be effective against tunneled traffic in certain situations, but did not present any concrete attack techniques [3].

In this paper, we propose a concrete TA attack on tunneled smart home traffic: Signature-based Tunneled Traffic Analysis (STTA), which aims to identify smart home devices in real-time simply based on a few network packets that are accessible in a small time window. For attackers, this real-time device identification capability is very beneficial from at least two perspectives. One is that some attackers may not be able to continuously eavesdrop network traffic in real environments, and thus compromising user privacy based on a limited number of packets has a practical value. The other is that attackers may immediately perform certain malicious activities such as dropping network packets for the correspondingly identified devices.

STTA targets tunneled traffic in a smart home environment that has mixed network packets from different devices. It takes a machine learning approach, and only utilizes the packet size and order information accessible to attackers over the tunneled traffic to identify smart home devices in real-time. In the training phase, STTA constructs n-gram signatures for each device individually based on its common network traffic, and uses those signatures as the features to train a machine learning model. In the testing or attacking phase, given any network packet from the mixed traffic of different devices, STTA attempts to classify different combinations of network packet sequences (i.e., candidate n-gram signatures) as belonging to certain devices, and chooses the classification with the highest confidence score as the final device identification result. We evaluate the effectiveness of STTA using a dataset of tunneled traffic of 14 smart home devices, showing that STTA can achieve an 83% real-time device identification accuracy.

Existing defense mechanisms against TA attacks in smart home environments either do not work on tunneled traffic or incur too much bandwidth overhead (Section 2). Therefore, we further propose a simple packet-size obfuscation mechanism that aims to defend against our STTA attack. This defense mechanism is based on adding uniform random noise. It achieves approximate differential privacy, meaning that it provides a formal guarantee of preserving an individual device's privacy and gives a quantitative measure of privacy loss. Meanwhile, it incurs a much smaller bandwidth overhead in comparison with other mechanisms such as traffic shaping. Our evaluation results show that this defense mechanism can decrease the accuracy of our STTA attack to around 25% and 10% (i.e., near random guessing) if an average amount of 15-byte and 40-byte per packet overhead is allowed, respectively.

We summarize the key contributions of this paper as follows: (1) we analyze existing smart home traffic analysis attacks and defenses for their strengths and limitations; (2) we propose a signature-based tunneled traffic analysis (STTA) attack to identify smart home devices in real-time, and demonstrate its effectiveness; (3) we propose a simple yet effective packet-size obfuscation mechanism to defend against our STTA attack.

In the rest of this paper, Section 2 reviews some background information and the related work; Section 3 defines the threat model; Section 4 describes and evaluates our STTA attack; Section 5 presents our defense mechanism and the defense evaluation; Section 6 discusses limitations and recommendations for our attack and defense; Sections 7 makes a conclusion.

2 BACKGROUND AND RELATED WORK

In this section, we review the background on traffic analysis and tunneling as well as related work.

2.1 Background

Traffic Analysis (TA) is a procedure that examines network traffic to understand the purpose of communication. TA is not necessarily harmful, and is indeed used in many legitimate systems such as in intrusion detection, intrusion prevention, and service quality monitoring systems [17, 31]. For attackers, TA can be used to compromise user privacy. Internet Service Providers (ISPs) may also use TA to differentiate network traffic [16], especially considering that the net neutrality regulations have been recently stopped in the U.S. [37].

Much of the existing work on traffic analysis is in the context of website fingerprinting (WF), where an attacker attempts to identify which website is being visited by a user based on the observed network traffic. There exist multiple website fingerprinting techniques such as deep packet inspection (DPI) and flow or packet based analysis. While the same or similar techniques can be used by attackers to analyze the IoT traffic, their effectiveness might be severely limited. For example, DPI is often computationally demanding, and may even not work at all on the contemporary IoT traffic which is largely encrypted [28].

By a tunneled environment, we mean that some traffic tunneling technique is in use to create a private network communication channel between two endpoints. This provides strong protection to the exchanged data even over a public network. There are many ways to tunnel traffic such as using a Virtual Private Network (VPN) or creating a Secure Shell (SSH) channel. In this paper, we consider that the IoT traffic is tunneled using a VPN; therefore, the packet size, direction, and time are the only metadata accessible to attackers.

2.2 Related Work

The main website fingerprinting technique applicable to our work is packet-based traffic analysis, which can still be successful in a tunneled IoT environment if properly designed. We review related website fingerprinting attacks and defenses, and then review related smart home TA attacks and defenses.

2.2.1 *Website Fingerprinting.*

In [9, 19, 30], researchers show that WF can be done even in the presence of advanced privacy enhancing technologies such as VPN or The Onion Router (TOR) browser. These attacks are similar to our attack from the perspective of having to analyze the tunneled traffic. However, they rely on some key features that are not easily extractable in a smart home environment with the mixed traffic of different devices. For example, Hermann et al. [9] used the packet size frequency distribution as a key feature while Draper-Gil et al. [19] and Wang et al. [30] used packet intervals as some of their key features to identify the websites. In a smart home environment with mixed traffic, obtaining packet size frequency distributions and packet intervals per device is challenging. Wang et al. proposed a WF technique with consideration to the multi-tab condition [29]. A multi-tab condition means that multiple websites are exchanging packets with a browser at the same time, which makes conventional WF methods that rely on traffic flows unable to accurately split the flows. The solution proposed in [29] was to develop algorithms to split the tabs into different flows so that existing WF techniques can be used. Their split algorithms are based on either a time gap between two websites or a machine learning algorithm that is trained to find the optimal split point between two websites. However, these splitting algorithms are not suitable for a mixed smart home environment because we cannot rely on a time gap between different IoT devices' traffic flows, and information such as session establishment cannot be used for training a machine learning classifier to differentiate the flows.

WF countermeasures have also been studied intensively in the last decade. For one example, Dyer et al. analyzed nine TA defense mechanisms and concluded that none of them were effective in preserving privacy without incurring large overhead [14]. For another example, Cherubin et al. observed that the most important component of a TA attack is the feature set, not the strength of the attacking classifiers [6]; therefore, correctly identifying and obfuscating the exploitable features could be the best defense. These

Table 1: TA attack classifiers in smart home environments and the relevant features used by them.

TA classifiers	Relevant features
Acar et al. [1]	Mean packet length, mean inter-arrival time, standard deviation in packet lengths, etc.
Sivanathan et al. [28]	Active volume, DNS interval, average packet size, mean rate, sleep time, number of servers, number of protocols, etc.
Apthorpe et al. [3]	Mean traffic volumes, number of packets for each flow, inter-packet intervals, etc.
Meidan et al. [26]	TCP sessions, number of sessions, sequence sizes of sessions, etc.
Our STTA	Packet size, packet order

observations in the WF defense research are helpful to us in designing our defense against TA in IoT environments.

2.2.2 Traffic Analysis in Smart Homes. There have been noticeable efforts in the arena of traffic analysis in smart homes. [1–3, 26, 28] provided some examples of successful TA attacks. In more details, attacks in [1] were able to not only identify devices of a smart home but also infer user activities; attacks in [28] exploited local traffic data as well as flow-based features such as active volume, DNS (Domain Name System) interval, sleep time and other features to classify IoT devices; attacks in [3] also used the DNS calls to help classify IoT devices. For example, if a Samsung server is contacted, it is very likely that the device is made by Samsung. Copos et al. analyzed network traffic to infer whether the house is occupied or not [7]. The authors studied Nest Thermostat and Nest Protect device traffic and could identify the transitions between home and auto away mode and vice versa with 88% and 67% accuracy, respectively, allowing attackers to identify when the home is occupied. Meidan et al. used TCP sessions to perform TA classifications [26]. All of these research projects focus on network traffic that is not tunneled, and use features that would not be available in a mixed and tunneled environment.

Kawai et al. [20] only used packet size and packet inter-arrival time (IAT) to identify devices. This method is scalable as it does not need protocol specification. However, their evaluation considered only a few devices; more experiments with a diverse set of smart home devices are needed for a realistic evaluation. Besides [20], all mentioned attacks were able to be very accurate due to the availability of highly relevant flow-based features, as summarized in Table 1. However, those features are not easily extracted from the tunneled and mixed traffic of different devices. Our STTA attack only uses the available packet size and packet order information.

From the smart home TA defense perspective, Apthorpe et al. proposed a solution that implemented a tunnel at the router level to shape the traffic of a smart home, making devices indistinguishable [3]. This solution is powerful because it hides the metadata and traffic rates. However, it is an expensive solution because a large amount of cover data is needed to shape the traffic properly. The authors showed that in their environment this solution needs around 104 GB extra data per month for high traffic devices. Some other issues with this solution include the possible delay to some packets for the purpose of traffic rate balancing and the possible fragmentation of some packets. These reasons make a naive traffic shaping solution undesirable.

Datta et al. did a follow-up work in which traffic shaping is applied at the device level as opposed to the router level, and the traffic shaper enforces a random instead of a constant traffic rate [8]. These improvements can help reduce the overall traffic bandwidth overhead, but they transfer the burden of the proper implementation of the solution to IoT device developers, raising many new questions such as “Do developers care enough about user privacy?” and “Are different device vendors willing to adopt this solution?”.

Apthorpe et al. proposed stochastic traffic padding (STP) as a lightweight mitigation to their flow-based inference attacks [2]. It is worth noting that their mitigation method considered only masking activity events that are triggered by users, not the periodic events. Our packet-based STTA attack would still work even if STP is implemented as STP would not change packet sizes for periodic events. Additionally, our defense masks both trigger and periodic events.

Another TA defense was recently proposed in [22]. It utilizes the notion of a smart community which geographically connects many smart homes to each other as a base to protect user privacy. The smart community hides the source of the smart home traffic by rerouting packets through different homes, obscuring the potential linkability of data that a traffic observer may derive. A clear limitation of this solution is that no protection will be provided if a smart home does not have some neighboring smart homes. In addition, some smart homes may fail to reroute the traffic, thus reducing the reliability of this community-based solution.

3 THREAT MODEL

Figure 1 illustrates the basic threat model that we consider in this paper. The owner of a smart home sets up a secure tunnel with a remote VPN server to protect the network traffic of IoT devices. The VPN server relays the traffic to and from the IoT services that are deployed by the device vendors for providing functions and capabilities such as cloud-based storage.

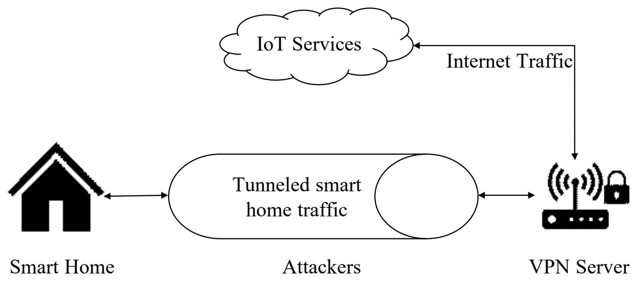

Figure 1: Illustration of the basic threat model.

Attackers that we consider are network traffic observers who can only observe the tunneled traffic. The attacker’s goal is to identify packets belonging to specific smart home devices in real-time. The attackers focus on identifying smart home devices because it is the first step in both the inference of user activities for compromising privacy, and the intrusion of devices for compromising security. Meanwhile, they aim to identify the devices in real-time because they may not be able to continuously eavesdrop network traffic in real environments, or may want to immediately perform certain malicious activities such as dropping or delaying network packets for the correspondingly identified devices. Even ISPs may identify devices for traffic differentiation and prioritization purposes [16] especially considering that the net neutrality regulations have been recently stopped in the U.S. [37].

We assume that attackers have knowledge about the set of the devices that are deployed in a smart home. This assumption is reasonable in many situations. For example, the smart devices purchased for homes in a managed community could be the same and the related information is publicly available. Attackers can build a labeled database of smart home device traffic, and train their classifier on the data produced by individual smart devices, as will be further explained in Section 4.

We assume that the network traffic for the smart home is tunneled. Over the tunnel, attackers can only obtain the packet size, time, and direction features. We do not assume that attackers can access the IoT devices or their local communication with each other. We do not assume that attackers can access IoT services of the device vendors or their communication with the VPN server. In other words, our focus is how the attackers may perform real-time device identification for each packet on the VPN tunnel, which is presumed to be sufficiently secure.

4 ATTACK

In our attack, we focus on IoT devices' traffic behaviors. Therefore, we designed a new attack to show the possibilities of TA attacks on tunneled traffic. We chose to use packet-based TA, as it is more applicable to a tunneled smart home environment. Flow-based TA would only work if the smart home has only one device. If the smart home has more than one device, flow-based and timing-based TA attacks cannot aggregate a traffic flow, which will inhibit their abilities to identify smart home devices properly. According to McKinsey's 2018 report on smart homes, 44% of smart homes have more than two devices [39], so flow based TA is unlikely to be effective. Packet-based TA is a good choice to perform attacks, as tunneling cannot hide the size of single packets. Even though these features are attainable for packet-based TA, mixed traffic makes it hard. For example, the work in [19] uses packet size distribution as a feature. The mixed and tunneled traffic makes attaining the distribution for each device challenging. Thus, there is a need to derive new features for a TA attack on mixed and tunneled smart home traffic. Figure 2 shows how extracting sequences of packets for individual devices is hard using current TA techniques as they traditionally rely on IP addresses to split the traffic into flows. VPN tunneling hides the IP addresses and other metadata, which makes tunneled packets indistinguishable in all features except size, time, and direction.

We want to build features that effectively capture a device's signatures. By signature, we mean a pattern of one or more consecutive packets that are frequently sent by a specific device. A device

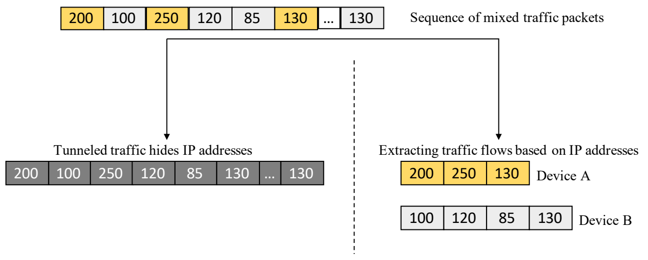
200
100
250
120
85
130
...
130
Sequence of mixed traffic packets
Tunneled traffic hides IP addresses
Extracting traffic flows based on IP addresses
Device A
Device B

Figure 2: Sequence of packets in a smart home with multiple devices. Different colors represent different IP addresses. Our STTA attack can perform TA on the tunneled traffic.

may have more than one signature. If a signature is encountered in real traffic, it is likely that the packets in the signature belong to the corresponding device. The usefulness of a specific signature for identification depends both on how frequently it is sent by the device, and how unique it is. If many other devices send the same signature packet sequence, then that signature is ambiguous, and will likely be less useful for identification. Since signatures are sequential packets, we chose to use the sizes of n sequential packets as our features.

We also need to be able to recognize these signatures in real traffic. In a smart home there could be many devices, so traffic is often mixed. This means that a signature for one device may be interrupted by packets from a different device. In order to recognize a signature in a mixed traffic environment, we need to aggregate groups of packets, and construct potential signatures. Then, we can use these signatures to do real time classification of which device a packet belongs to.

4.1 Design of Signature-based Tunneled Traffic Analysis (STTA) Attack

Our attack consists of two general phases: training and packet identification (Figure 3). During training, we attempt to determine the signatures made by devices. Traffic for each device is isolated, and organized into groups of n sequential packets, or n-grams. A classifier is then trained to associate n-grams with a particular device. Testing is somewhat less straightforward, as traffic may be mixed. This means that the packets in each device's signatures may no longer be sequential. In order to overcome this problem, we look at many packets surrounding the current packet, and construct many candidate n-grams. Each one of these n-grams is given to our classifier, and the candidate n-gram for which the classifier has the highest confidence in its prediction is chosen, and the device is identified.

We assume that tunneled traffic was implemented as a defense tool, so our attack only uses packet size and order. We show that even with these limited features, we can successfully identify packets in real time with STTA.

4.1.1 Training. We need to build a classifier that is able to learn device signatures from a traffic trace. In order to do this, we need to train it on features that are capable of representing a signature. Working from the insight that IoT devices tend to have periodic events where they send a number of packets in a short time interval to keep their servers continuously updated, we chose to examine

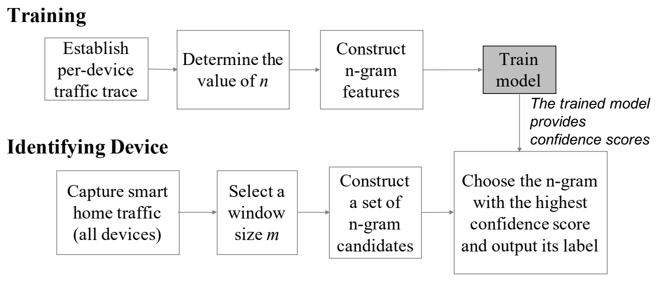
Training
Establish per-device traffic trace
Determine the value of n
Construct n-gram features
Train model
The trained model provides confidence scores
Identifying Device
Capture smart home traffic (all devices)
Select a window size m
Construct a set of n-gram candidates
Choose the n-gram with the highest confidence score and output its label

Figure 3: Attack design of STTA.

the group of n packets surrounding every packet in a device's traffic trace (n-gram). This n-gram is a potential signature for a device.

Obviously, choosing the value of n correctly is crucial to the accuracy of the classifier. We want n to be the optimal signature size, such that the signatures of size n are as useful as possible. The usefulness of a signature depends both on how frequently it is sent by this device, and on how unique it is among devices. For this reason, term-frequency inverse-document-frequency (tf-idf) is a good measure of the usefulness of a specific signature. This represents the ratio of the number of times a signature appears in the traffic trace from a particular device to the total number of times the signature appears from any device. During experimentation, we found that scoring with normal tf-idf gives disproportionately large scores for large values of n. This is because a large n tends to have thousands of unique packet combinations, the majority of which are not useful and will not appear in practice due to mixed traffic. Although the tf-idf scores for these signatures are very low, the massive number of signatures present still impact the sum. Sublinear tf-idf solves this problem by replacing term frequency with log(term frequency) + 1. Changing to sublinear tf-idf ensures that the very large amount of relatively useless signatures for large n do not disproportionately affect the score.

In order to choose n, we run sublinear tf-idf to score all of the potential signatures for a certain signature size. The sum of the tf-idf scores for all of these signatures gives a measure of how useful the signatures of this size are for identifying devices. To choose n, we simply pick the signature size that yields the largest sum of sublinear tf-idf scores. This is shown in Algorithm 1.

Algorithm 1 Determine the value of n

Input traffic traces (T) for all devices in a smart home
Output n

```
1:  $T_i$ = Traffic trace for $i^{th}$ device
2:  for $n$ = 1 to MAX_N do
3:      $score_n = 0$
4:      for $i$ = 1 to $\|T\|$ do
5:          $signatures_{i,n}$ = set of all sequential $n$-grams in $T_i$
6:          scores = TF-IDF($signatures_{i,n}$, $T$)
7:          $score_n$ += Sum(scores)
8:      end for
9:  end for
10: return $n$ with largest score
```

After n has been chosen, n-gram features can be constructed for each device by collecting all sequences of n sequential packets in each device's traffic trace. The features used by the classifier will be the sizes of the n sequential packets. Then, a machine learning classifier can be trained on these features. Thus, this classifier will be able to predict which device a certain n-gram belongs to.

4.1.2 Identifying Smart Home Devices in Real Time. Using the classifier to predict the current device is not straightforward, however, because of the mixed traffic often present in smart homes. Our model needs to account for the fact that there may be packets from other devices in between the signature packets from one device. To overcome this challenge, we need to consider different arrangements of packets, in the hope of one arrangement being of the signature for a specific device.

To do this, we construct a window for each packet which consists of the group of m surrounding packets. Then we form all possible combinations of n-grams from the window that include the current packet. The value of m should be chosen based on the analysis of smart home devices' traffic behaviors, or based on experimentation. We found that larger values of m tend to have better results, but take longer to classify. Further, there is a diminishing return on accuracy as m gets larger. This makes sense, as most smart home devices tend to have small periodic packet sequences sent in a short period of time [16]. Thus, the best m will likely not be significantly larger than n.

Each of these candidate n-grams is then given to the previously trained classifier, which outputs the predicted device and a confidence in that prediction. The label of the n-gram with the highest confidence score is chosen to be the identified device for this packet. Algorithm 2 describes this identification process for $n = 2$, which is the best value of n for our dataset, as detailed in Section 4.3. The '$\times$' symbol represents the Cartesian Product operation. The '$\bigcup$' symbol represents the set union operation.

Figure 4 illustrates this algorithm if n is 2 and m is 7. For each of these 2-grams, we use the classifier to calculate the probability of it belonging to any individual device. The number of 2-grams generated depends on the size of window m. For example, using a window of size 7 will result in 6 different combinations of 2-grams. The classifier's prediction with the highest confidence score will be used to identify the device.

Algorithm 2 Classify a Single Packet Based on *2*-Gram

Input current packet p_i
Output device class of p_i

```
1: $W = \{p_{i-\lfloor m/2 \rfloor}, \ldots, p_{i-1}\} \times \{p_i\}$
2: $W = W \bigcup \{p_i\} \times \{p_{i+1}, \ldots, p_{i+\lfloor m/2 \rfloor}\}$
3: for every 2-gram $w_j$ in $W$ do
4:     $(label_j, confidence\_score_j) = classify(w_j)$
5: end for
6: return $label_j$ with the highest confidence score
```

4.2 Dataset

We used a publicly available dataset of common IoT devices to train and test our algorithm. The dataset has 20 different IoT devices and it was collected over 23 days [28]. The authors set up a smart environment and captured all the traffic generated from the IoT devices using tcpdump [33]. The advantage of this dataset is that it has a very diverse set of IoT devices. The list of IoT devices with more details can be found in [28]. We trained on traffic from October 2nd-5th, and tested on traffic from October 6th-10th.

Out of the 20 IoT devices in the dataset, four devices were only active for a few days. Another two devices were only active when they were triggered. Both of these situations were not helpful in training as they did not generate periodic events, so they were removed. One major challenge in IoT device identification is that IoT devices manufactured by the same vendor tend to have similar, if not identical, network packet sizes because they share similar hardware or software. Authors in [16, 27] demonstrated the similarities among devices from the same vendors and the difficulty in identifying those devices. One solution to this problem is to group these similar devices to fall into the same classification category, as done in [24]. We take this approach in our attack, and form 3 groups of devices with the same vendor. In total, we have 11 IoT device classes, 8 of which are distinct devices, and 3 of which are vendor classes, each with 2 distinct devices, for a total of 14 devices. Table 2 shows the devices and target classifications used in our experiment.

In order to simulate a tunneled traffic environment, we removed all local traffic and other metadata such as IP addresses and protocols. Next, we needed to isolate individual device traffic in order to train our classifier. In a real world situation, an attacker would likely not have access to actual labeled data of the victim smart home. Instead, they would have to train on their own devices or obtain a publicly available dataset. However, in our experiment, we only had one dataset to work with. Therefore, we needed to extract the data that the attacker would be able to obtain from their own devices or dataset.

Table 2: Devices used in our experiments. Devices from the same vendor were grouped into the same class.

Classes	Devices
Class 1	Amazon Echo
Class 2	Drop Camera
Class 3	Light Bulb
Class 4	Samsung Camera
Class 5	Inteson Camera
Class 6	HP Printer
Class 7	SmartThings
Class 8	Triby Speaker
Class 9	**Netmato vendor**: Netmato Camera, Netmato Weather Station
Class 10	**Withings vendor**: Withings Smart Scale, Withings Sleep Track
Class 11	**Belkin vendor**: Belkin Motion Sensor, Belkin Switch

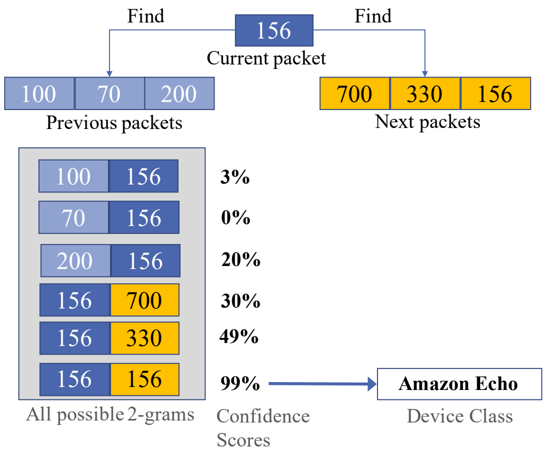

Figure 4: *n*-gram construction and identification when $n = 2$ and $m = 7$. The numbers in the rectangles are packet sizes.

It is clear by now that IoT devices of smart homes have some distinct traffic behaviors when traffic is not tunneled as reviewed in Section 2.2. To estimate the possibility of finding signatures among packet sizes in the dataset, we analyzed the uniqueness of single packets and pairs of packets. For single packets among all devices, the uniqueness of the single packets ranged from 10% to 20% per device. For the pairs of packets, the uniqueness rate rose to be between 60% and 80% per device.

The traffic in the dataset was mainly one of two types, either periodic or trigger-based. Trigger-based traffic is traffic generated by events where a user interacts with the devices. For example, a user could send a request to a smart camera to stream live videos of their house. In contrast, periodic traffic is when devices send packets throughout the day to update some information or simply "check in" with the manufacturer's server. Many IoT devices send a small amount of data at constant time intervals, as explained in [16, 28]. These events generate the majority of the traffic. This periodic data is independent of interactions between the device and the user, and thus should be the same regardless of where the device is deployed.

In order to make our attack focus only on periodic events, we contacted the dataset authors and asked a few questions regarding how the dataset was generated. We asked if devices were intentionally triggered and if there were times where users tended to be around the devices more than others. The authors responded that intentional interactions with devices were minimal and mostly towards the beginning of the data collection, and that times spent around devices varied. This tells us that the dataset has some user interactions but mainly in the beginning of the collection. To avoid triggered activity, we only used the traffic traces towards the end of their collection. Additionally, we removed the non-periodic trigger-based traffic using Fast Fourier Transform (FFT) [23]. FFT is a well-known algorithm used in signal processing to de-noise signals. FFT is an efficient algorithm that uses the Discrete Fourier Transform (DFT) to transform the signals from the time domain to the frequency domain. We used this to keep only events with high frequency rates. Low frequency rates are most likely to refer to specific trigger based behavior, so they were removed.

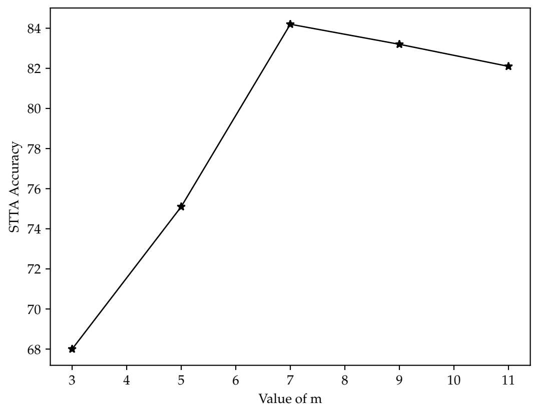

Figure 5: Comparison of STTA accuracy for different window sizes.

After removing the user-triggered traffic, we noticed that some devices sent significantly more periodic packets than others, Drop Camera for example. In order to solve this class-imbalance issue, we used the Synthetic Minority Over-sampling Technique (SMOTE) [5]. SMOTE works by generating extra, representative synthetic training samples for under-represented classes. This helps to remove machine learning classifiers' bias towards predicting over-represented classes.

4.3 Experimental Setup

We used a variety of available tools to implement STTA. All scripts were written in Python 3. Feature extraction was performed using TShark, which is the Wireshark Network Analyzer [34]. To implement machine learning classifiers, we used the Scikit-learn library, which is a poplar library for machine learning projects. We also used Pandas library for data analysis and data preparation [25].

We experimented with a variety of classifiers, such as K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and RandomForest [4], and eventually chose RandomForest as we found that it performed the best. This classifier is suitable for a couple of reasons. First, it performs well for multiclass classification, which is desirable since we have 11 different device classes. Second, RandomForest is very good at avoiding overfitting, since it averages the outputs of many different decision trees.

Throughout the experiment, our goal was to have a realistic test environment. We treated data as it could be seen by an ISP or a network observer, and only used the features of packet size and order on outgoing traffic. IoT devices often generate traffic more than receiving due to their nature of sensing from the physical world and transforming to a digital format; thus outgoing traffic is more likely to be useful for identification.

Choice of n and m. In order to determine n for our set of devices, we implemented Algorithm 1. The signature size with the largest sum of tf-idf scores was $n = 2$. Thus, we trained our classifier to associate all of the 2-grams in the traffic trace with their corresponding devices. We did this for every 2-gram in all devices' periodic traffic. The classifier is then able to try and identify which device any 2-gram of packet sizes belongs to.

To find the best window size to use, we wrote a script to test our classifier on various values of m. Figure 5 illustrates the effects of choosing different values of m. We found that as m increased, testing accuracy increased, until it peaked around 7. Values of m larger than 7 led to slightly lower accuracy scores. Therefore, we chose 7 as the best option for m as it gives the best results.

Choosing 7 for m means that we will build 2-grams using the three previous packets and three subsequent packets with the current packet, resulting in 6 possible combinations. Then, every combination will receive a confidence score and the highest score will identify the device that sent the packet, as shown in Figure 4.

4.4 Results

In our experiments, we trained a classifier using all 2-gram's from the traffic traces of the dataset on October 2nd-5th of the dataset. We then tested its ability to identify the sending device of a packet on October 6th-10th. For testing, we still simulated tunneled traffic by removing all features except for packet size. We did not remove the triggered traffic for testing in order to make the attack more realistic.

To analyze the results of our classifier we used accuracy, precision, recall, and F1 score. Accuracy is a very widely used performance measure that represents the ratio of correctly predicted observations to the total observations. For each device, precision measures the proportion of times packets were correctly identified over the total number of all predictions to that device. For each device, recall measures the proportion of times the device was identified correctly to the total number of times this device sent packets. Finally, the F1 score for each device represents a balance between precision and recall. For multiclass classification, the reported value for each of these metrics is averaged across all devices. The formulas for each of these metrics is presented below, with TP meaning true positive, FP being false positive, TN being true negative, and FN being false negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

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

The results of our experiments are shown in Table 3. The classifier was able to consistently identify the correct device 83% of the time. This accuracy is fairly consistent regardless of the day it was

Table 3: Evaluation results of identifying devices by STTA.

Tested on	Accuracy	Precision	Recall	F1 Score
October 6th	82%	83%	82%	82%
October 7th	82%	82%	81%	81%
October 8th	83.4%	85%	83%	83%
October 9th	83.9%	85%	84%	83%
October 10th	84.3%	85%	84%	84%
Average	83.1%	84%	82.8%	82.6%

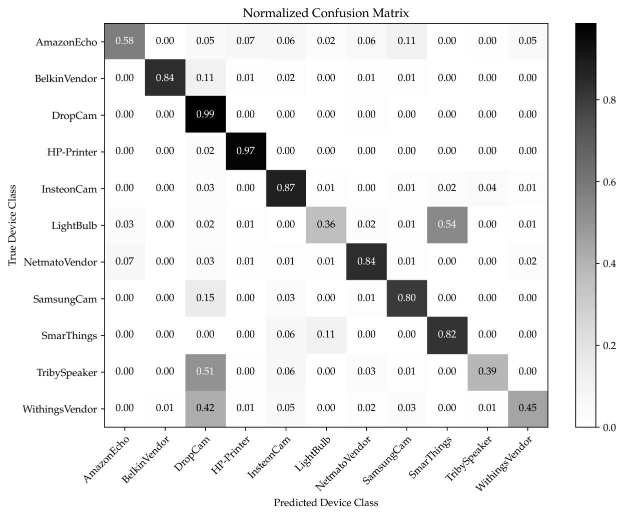

Figure 6: Confusion matrix of October 10th using STTA.

tested on. The number of packets identified varied on different days, with the average being 200 thousand packets per day. In our experiments, some classes were more recognized than others, which can be shown by a confusion matrix. Figure 6 shows the confusion matrix of STTA results when performed on October 10th of the dataset. Each row in the confusion matrix represents the ground truth of device classes and each column shows the predicted classes using STTA. Each cell in the matrix represents the percentage of classification of each class. STTA predicted some classes more accurately than others. One reason for this is that devices with more packets tend to be correctly classified by STTA more than devices with fewer packets. This occurred despite using SMOTE oversampling in training. For example, Drop Camera, which generated the most packets, was recognized almost all of the time, while packets from LIFX Light bulb device, which occurred infrequently, were the least recognizable with an average accuracy of 36%. Another interesting finding was the high rate of Triby Speaker packets being misclassified as Drop Camera packets. We do not know the reason behind the similarities in packet sizes between these two devices, as they were manufactured by different vendors. One potential reason is the use of similar libraries.

To measure the completeness and quantity of our STTA, we look at the recall results. High recall means that our STTA returned most of the relevant results. Another interesting finding was that the STTA classification accuracy is lower for more complicated devices. For example, Amazon Echo has the most diverse functionality out of the tested devices. Users of Amazon Echo can ask questions, play music, or set reminders, while other devices tend to have limited functionalities like Drop Camera, which is either recording or streaming. Due to its large set of functionalities, Amazon Echo has a very diverse set of packet sizes which makes it relatively harder for STTA to correctly classify.

We looked into the potential reasons for misclassification. The reasons were likely due to the situations that packet sizes were not unique, the value of m was not large enough to catch the pair of packets from the same device, or packets were generated from user activities. The dataset we used did not have labels for user activities and recall that we removed the non-periodic trigger-based traffic in the training data using FFT. We manually analyzed a set of 500 misclassified packets chosen randomly. User activities called for nearly half of the misclassification as we could not identify the signatures in the training data. The uniqueness of the packet size and the value of m called for nearly a third of the misclassification. We could not identify the reasons for the rest misclassifications.

It is worth mentioning that the random guess for device class identification is below 10%. In our experiments, the lowest percentage for a class was 36%. The results show that device signatures can be used to identify packets even if traffic is tunneled and mixed.

4.5 Experiments in a Noisy Environment

We also want to evaluate our STTA if non-IoT devices and IoT devices coexist in the same network. Non-IoT devices can be personal computers, smart phones or tablets. Non-IoT devices are often easily identifiable because users may only use their personal devices in a small portion of the day, whereas IoT devices keep communicating all the time with periodic traffic, as explained in Section 4.1. Also, non-IoT devices tend to receive more packets than they send. Garrett explained key differences between IoT and non-IoT devices in terms of network behaviors [16].

We performed the experiments again, but this time left in traffic the packets from non-IoT devices. There were several non-IoT devices in the testbed, such as laptops, mobile phones and an Android tablet. Our STTA attack still performed well in this new experiment, achieving on average 78% accuracy. This is likely because the average size of packets from non-IoT is dramatically bigger than that of IoT devices and our classifier was able to recognize this. In the dataset, the average packet size of IoT devices was less than 160 bytes while non-IoT devices had an average packet size of 699 bytes. Another potential reason why the STTA accuracy did not decrease notably is because the proportion of non-IoT traffic is not very high; the smart home devices sent much more traffic than the non-IoT devices, likely due to their constant periodic communication.

5 DEFENSE

In this section, we present a packet-size obfuscation mechanism that uses random noise addition to protect against device identification attacks such as STTA over tunneled network traffic. The goal of random noise addition is to obscure the key feature, packet size, that still remains visible in tunneled traffic. We first present a background on differential privacy. We then analyze a commonly used differentially private defense mechanism that is not suitable for our goal, and present our defense mechanism with proof and experiments to show that it achieves approximate differential privacy.

5.1 Defense Design

Adding noise occurs at the smart home endpoint of the tunnel. Figure 7 illustrates where the noise is added and removed. The smart home end of the tunnel will indicate the amount of padding and encrypt the packet. The VPN end can then decrypt the padded packet and remove the padded bytes.

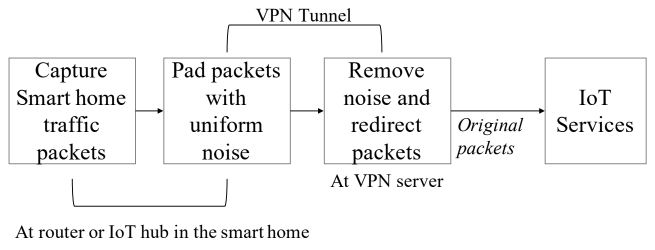
VPN Tunnel
Capture Smart home traffic packets
Pad packets with uniform noise
Remove noise and redirect packets
Original packets
IoT Services
At VPN server
At router or IoT hub in the smart home

Figure 7: Noise addition and removal.

5.2 Differential Privacy

Differential privacy (DP) was first proposed to quantify individuals' privacy loss in statistical queries on traditional databases, and it is the state of the art metric on measuring the privacy guarantees of randomization algorithms [12]. DP algorithms have become a trend in the privacy research community, and have also been adopted by companies such as Apple [32] and Google [15].

Formally, an algorithm $\mathcal{A}$ is ϵ differentially private if for any two databases D and D' differing at most in one row and for any subset $\mathcal{S}$ of elements from Range($\mathcal{A}$), Inequality (1) is satisfied [10].

$$Pr[\mathcal{A}(D) \in \mathcal{S}] \leq e^{\epsilon} Pr[\mathcal{A}(D') \in \mathcal{S}] \quad (1)$$

If this inequality is satisfied, then the algorithm's output is said to be ϵ indistinguishable, and it is hard for an attacker to tell if any individual is present or not based on the algorithm's output. Here ϵ is the privacy deficit, and gives a quantitative measure of an individual's privacy loss. Lower values of ϵ correspond to stronger individual privacy. DP is desirable for our defense because it means that attackers will not be able to accurately associate a packet with any individual device.

Approximate Differential Privacy: In [11], Dwork et al. present an approximate version of differential privacy. An algorithm $\mathcal{A}$ is said to be (ϵ, δ) approximately differentially private if Inequality (2) is satisfied.

$$Pr[\mathcal{A}(D) \in \mathcal{S}] \leq e^{\epsilon} Pr[\mathcal{A}(D') \in \mathcal{S}] + \delta \quad (2)$$

Approximate differential privacy is a relaxed version of normal differential privacy. The addition of the δ term allows an algorithm $\mathcal{A}$ to not be ϵ differentially private for some proportion of inputs. Therefore, δ can be interpreted as the proportion of the probability distribution that does not fall into the range allowed under ϵ differential privacy. Smaller values of δ correspond to an increased level of individual privacy.

Our mechanism achieves approximate differentially privacy, meaning that, within parameters (ϵ, δ), it is hard for an attacker to distinguish one device's packets from those of another device based on the packet size. While differential privacy is stronger and more desirable than approximate differential privacy, achieving the latter is more practical and can still be effective for preserving privacy.

5.3 Obfuscation Algorithms

Consider an obfuscation algorithm $\mathcal{A}$ which pads network packets with bytes of random noise:

$$\mathcal{A}(s) = s + \theta \quad (3)$$

where s is the original packet size and θ is a random amount of noise sampled from the probability distribution function $f_\theta(x)$. For any input x, $f_\theta(x)$ represents the probability that an amount of noise equal to x (θ = x) will be chosen. If a packet will be larger than MTU after adding noise, it will be padded to the MTU size instead. This is very unlikely to happen for periodic traffic, because the packet sizes are small. Triggered events with size of MTU are not unique, so capping the size at the MTU should not decrease the defense's effectiveness.

The privacy protection of the system depends on the ability of $\mathcal{A}$ to make packets originating from different devices indistinguishable from each other. First, we describe a more traditional approach using Laplacian noise, and discuss its limitations. Next, we describe a better defense mechanism based on uniform random noise, and show it satisfies (ϵ=0, $\delta=\frac{\Delta s}{w}$) approximate differential privacy.

5.3.1 *Laplacian Noise.*

A traditional method for achieving differential privacy is by adding noise according to the Laplace distribution: $f_\theta(x) = Lap(\frac{\Delta s}{\epsilon})$, where Δs is the sensitivity, i.e., the maximum difference between neighboring databases. This method is well studied, and achieves ϵ differential privacy [13]:

$$f_\theta(x) = Lap(\frac{\Delta s}{\epsilon}) = \frac{\epsilon}{2\Delta s} \exp(-\frac{\Delta s\,|x|}{\epsilon}) \quad (4)$$

In the context of traffic analysis, Δs is the maximum possible difference in packet size in a network. While this method does achieve ϵ differential privacy, it has the major disadvantage of allowing negative values for θ. On average 50% of packets will need to be reduced in size. In order to achieve this size reduction without the loss of information, network packets will have to be fragmented. However, fragmentation is undesirable for two reasons. First, it means that noise addition must be implemented at the device level because IPv6 does not support packet fragmentation at the router level [35]. Second, it would increase the number of packets sent, and possibly cause delays. This makes a Laplacian noise implementation less practical, and not suitable for many applications.

5.3.2 *Uniform Random Noise.*

To solve the implementation problems associated with Laplacian noise, we propose to use uniform random noise. While this method only achieves approximate differential privacy, it is much more practical. Furthermore, in our experiments shown below, uniform noise actually outperformed Laplacian noise, resulting in a slightly better protection against device identification.

For uniform noise, a random number of bytes between 0 and w will be added. The probability distribution function is given by:

$$f_\theta(x) = \begin{cases} \frac{1}{w} & 0 \leq x \leq w \\ 0 & otherwise \end{cases} \quad (5)$$

On average, this will result in $\frac{w}{2}$ bytes being added to each packet. This method has the benefit that the packet size will never have to decrease, eliminating the need for fragmentation. It achieves (ϵ, δ) approximate differential privacy with $\epsilon = 0$ and $\delta = \frac{\Delta s}{w}$.

Proof. From [18], an algorithm $\mathcal{A}$ achieves approximate differential privacy if there exists a positive constant c_b to satisfy

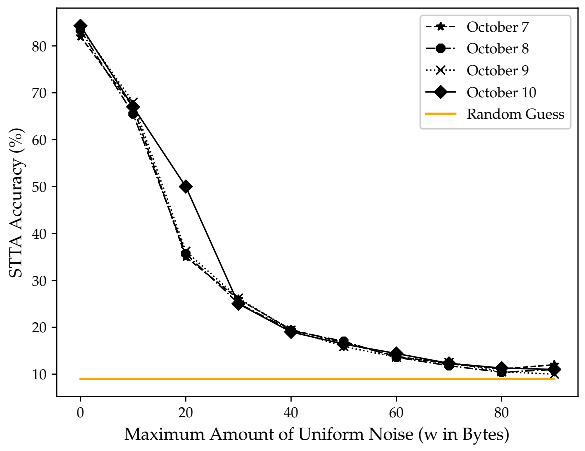

Figure 8: Effectiveness of using uniform noise as the defense mechanism.

Inequality (6):

$$\sup_{\hat{\sigma} \in [-\sigma, \sigma],\, f_\theta(x) \neq 0} \frac{f_{\theta+\hat{\sigma}}(x)}{f_\theta(x)} \leq c_b \tag{6}$$

Further, ϵ and δ are given by Formulas (7) and (8) [18]:

$$\epsilon \leq \ln c_b \tag{7}$$

$$\delta \leq \oint_{\Phi^0} f_\theta(x+\sigma)\, dx \tag{8}$$

Here, σ is the distance between adjacent input vectors. In our case, this is bound by the maximum difference in packet size in the network and $\sigma = \Delta s$, while Φ^0 is the zero point set of f_θ: $\Phi^0 = \{x \mid f_\theta(x) = 0\}$. It is easy to see that the condition in Inequality (6) is satisfied for uniform noise:

$$\sup_{\hat{\sigma} \in [-\sigma, \sigma],\, f_\theta(x) \neq 0} \frac{f_{\theta+\hat{\sigma}}(x)}{f_\theta(x)} = \frac{\frac{1}{w}}{\frac{1}{w}} = 1 \leq c_b \tag{9}$$

To find the strongest level of approximate differential privacy guaranteed by our algorithm, we derive the values of ϵ and δ using the lower bound of equation (9), $c_b = 1$. Therefore, from Formulas (7) and (8),

$$\epsilon = \ln 1 = 0 \tag{10}$$

$$\delta = \oint_{\Phi^0} f_\theta(x+\sigma)\, dx = \int_{-\infty}^{0} f_\theta(x+\Delta s)\, dx = \frac{\Delta s}{w} \tag{11}$$

and $\mathcal{A}$ satisfies $(0, \frac{\Delta s}{w})$ approximate differential privacy. □

5.3.3 Evaluation of the Defense. We evaluated our defense on the same dataset used in Section 4. We added a uniformly random amount of bytes between 0 and w to each packet, and measured the decrease in accuracy of the STTA attack for various values of w. This is shown in Figure 8. In our attack, STTA worked best with a RandomForest classifier and we used the same for the defense evaluation. It is worth noting that we also tested STTA with KNN and SVM classifiers and they showed no robustness to our defense.

From our results, even a very small amount of noise was able to obscure the packet size enough to result in a lower accuracy. Choosing w = 30 already decreased the accuracy by nearly 60%, which is only an average of 15 bytes added per packet. Increasing w to 80 makes the accuracy drop to near random guess, with an average of 40 bytes added per packet. Table 4 shows the amount of overhead corresponding to different values of w for different days.

Obfuscation via uniform random noise was very effective in reducing the accuracy of our STTA classifier. One reason that this method is so effective is that smart home devices tend to send a lot of packets with a small size. In the dataset we used, 97% of packets are less than 160 bytes, excluding local communication packets. Thus, adding only a little bit of noise already makes it much harder to distinguish devices.

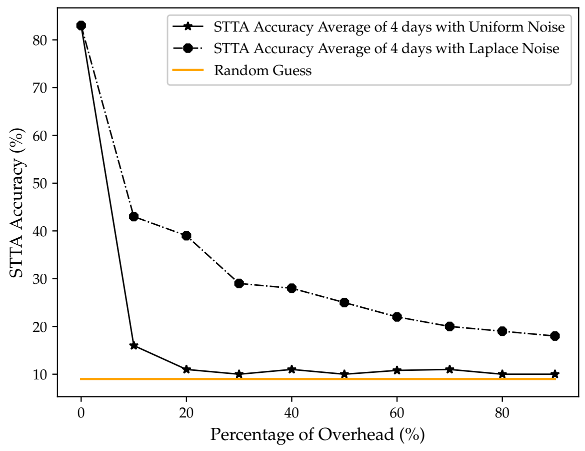

Figure 9: Comparison between Laplace noise addition and uniform noise addition.

In our experiments, uniform random noise actually outperformed Laplacian noise, resulting in a lower accuracy with less overhead. Figure 9 shows a comparison between the Laplace adding noise mechanism and the uniform adding noise mechanism. Uniform noise addition reaches the random guess level which is good in terms of protecting privacy with about 20% overhead. On the other hand, Laplace method needs at least around 60% of overhead to lower STTA accuracy to around 20%. The overhead percentage represents the amount of padding bytes out of the total bytes sent.

Table 4: Overhead percentage corresponding to different values of w for different days.

w (bytes)	Oct 7	Oct 8	Oct 9	Oct 10	Avg
10	1.4%	2.8%	3.3%	3%	2.6%
20	3%	6%	6.9%	6%	5.4%
30	4.6%	9%	10.6%	10%	8.6%
40	6.2%	12.3%	14%	13.8%	11.5%
50	7.9%	15.5%	18%	17.4%	14.7%
60	9.5%	18.7%	21.6%	20.9%	17.6%
70	11%	21.9%	25%	24.5%	20.6%
80	12.7%	25%	29%	28%	23.6%
90	14.3%	28.2%	32.6%	31.6%	26.6%

This may be surprising, as Laplacian noise is the standard noise adding mechanism in many differentially private models. However, this makes sense through analysis of the traditional goal of differential privacy. In traditional applications, differential privacy aims to keep the utility of the released data while preserving privacy. For our application, however, we have a different goal: to release as little information as possible to attackers while preserving privacy. From this perspective, it makes sense that uniform noise achieves much better results. When an attacker sees a packet of a certain size, the probability that any device generated a packet of that size is identical, assuming the device is capable of producing packets of that size. This is due to the uniform random distribution that has the same probability at all points, and is further shown by the algorithm being $\epsilon = 0$ approximately differentially private, meaning that there is not any privacy loss for some proportion of outputs. Although it does not achieve this $\epsilon = 0$ all of the time, when a packet is seen, and has a size capable of being generated by multiple devices, the probability of any one of those devices sending a packet of that size is the same, meaning that it is very hard for the attacker to gain any information regarding which device the packet came from. In the IoT environment, where many packet sizes are small, uniform noise addition is especially good at hiding packet size information even when a relatively small number of bytes are added. This analysis reveals that when it is possible that a packet came from different devices, uniform random noise is a superior obfuscation method to Laplacian noise, and releases very little information regarding which device the packet came from.

6 DISCUSSION

6.1 LIMITATIONS AND FUTURE WORK

One assumption that our attack relies on is that an attacker will know the devices in the target smart home, and has access to either a copy of these devices, or a dataset of the same devices' traffic to train on. It is entirely possible that an attacker would be able to figure out what devices were present in a smart home and then build or find training data specific to only those devices. However, the attack would be more versatile if instead, the classifier were trained on data for a wide variety of devices, and would then be able to identify the packets and devices within a certain smart home without prior knowledge of what devices were present. This is another possible direction of future work. If this were the case, STTA would easily be extended to allow for high level device identification as opposed to real-time per-packet device identification by aggregating the results of individual packet identifications, and analyzing which devices are most likely to be present in the smart home.

Uniform random noise is a defense mechanism which balances the need for privacy protection with a low amount of noise addition. In the current version of our defense mechanism, we considered all the smart home devices as equally important. However, in certain environments, some smart home devices are more important than others. In this case, our defense mechanism could be refined to weight smart home devices according to their importance. By doing this, network overhead will be notably less than the overhead of our current defense mechanism.

Our defense mechanism is deployed on two places: the router or the IoT hub in the smart home and the VPN server. These two ends are trusted. If an attacker gains access to either end, our defense mechanism would not be effective. Other works have suggested using one VPN server for multiple smart homes, making it difficult to link certain devices to specific smart homes [3]. This is out of the scope of this paper, and our solution instead allows the trust to be shifted from network observers and ISPs to a single VPN connection.

One important realization from our STTA attack is that even a simple machine learning model using only packet size and order was able to accurately identify smart home devices. In our experiments, we assumed a closed world scenario in which an attacker has knowledge of the devices existing in a smart home. Without accurate knowledge of the set of devices, attackers will not be able to create a training dataset. Even with a closed world situation, the number of smart home devices is a factor affecting the accuracy of our STTA classifier. As the number of devices increases, the accuracy will likely decrease. However, we tested our classifier on 14 smart home devices, which is a relatively high number, as current most smart homes most often do not have more than 10 devices [39].

A logical next step for attacks on tunneled smart home data would be user activity inference. This would attempt to identify the device and further attempt to identify what action triggered the device, what mode the device is currently in (e.g., home, away, locked, unlocked, etc), or even device information, such as the OS version. There have been previous works in this area, but they only consider untunneled traffic [1]. Thus, one promising area of future work would be to investigate whether activities inference is possible even with tunneled traffic. In this work, we did not try to infer user activities in smart homes because the dataset we used did not have labels for the users' activities.

One ongoing challenge for all traffic analysis techniques, whether in website fingerprinting or smart home device identification, is maintaining an up-to-date dataset. Keeping a training set fresh is important for many reasons. For example, a firmware update to smart home devices might change packet sizes or other network behaviors, which could result in device misidentifications. One way to make an attack resilient against these updates in devices is to use as simple a set of features as possible, so that the features are less likely to change with new updates. In our work, we used only packet size and ordering, which helps mitigate the effect of traffic changes due to software updates. Future work could investigate more directly the impact of device updates, and how to keep a resilient training set that mitigates impacts of updates.

6.2 RECOMMENDATIONS

We recommend smart home users to use at least some sort of protection such as a VPN service because VPN alone can protect users against many traditional TA intruders. We recommend that IoT hubs or routers and VPN vendors implement our obfuscation mechanism to further help protect smart home users against our STTA or similar TA attacks. In 2016, specific regulations were acted on in California to protect people who use smart meters in their houses [25]. We can see some similarities between smart meter and smart home privacy risks. For example, the risk of smart metering profiling includes what devices a user has, usage times, and user behaviors.

We recommend regulators to take actions similar to the smart metering situations, so that smart home users will be at least asked for consent before their data is analyzed and shared.

7 CONCLUSION

As the number of smart homes and devices continues to grow, so does the potential for privacy violations. Traffic analysis attacks are already capable of revealing private information of smart home activities for untunneled traffic. Simply, tunneling smart home traffic can be a very effective countermeasure to existing TA attacks. In this work, we showed that even tunneling network traffic is not enough to protect privacy.

Even with only packet size and order being visible, our STTA attack was still able to correctly identify the device with an 83% accuracy. This was possible because each device often has unique signature packet sequences, and STTA was able to identify and classify based on these signatures. This device identification is a major privacy violation, and is the first step towards other TA attacks such as activity inference.

We showed that adding even a small amount of uniform random noise to tunneled smart home traffic is enough to mask packet size and effectively protect against our STTA attack. Our defense mechanism does not require fragmentation, meaning that it can be implemented with IPv6. Overall, it is clear that privacy violations continue to be a major threat in the smart home environment. We hope this work encourages the smart home industry to be aware of our STTA attack and potentially adopt our defense mechanism to better protect user privacy.

ACKNOWLEDGMENT

We thank reviewers for their valuable comments. This research was supported in part by the NSF grant OIA-1936968. Ahmed Alshehri is supported by the Saudi Arabian Cultural Mission (SACM).

REFERENCES

[1] Acar, A., Fereidooni, H., Abera, T., Sikder, A. K., Miettinen, M., Aksu, H., and Uluagac, A. S. Peek-a-boo: I see your smart home activities, even encrypted! In *arXiv preprint arXiv:1808.02741* (2018).

[2] Apthorpe, N., Huang, D. Y., Reisman, D., Narayanan, A., and Feamster, N. Keeping the smart home private with smart (er) iot traffic shaping. In *Proceedings on Privacy Enhancing Technologies, 2019(3)* (2019).

[3] Apthorpe, N., Reisman, D., Sundaresan, S., Narayanan, A., and Feamster, N. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. In *CoRR, abs/1708.05044, 2017* (2017).

[4] Breiman, L. Random forests. vol. 45.

[5] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. In *Journal of artificial intelligence research* (2002).

[6] Cherubin, G. Bayes, not naïve: Security bounds on website fingerprinting defenses. In *Proceedings on Privacy Enhancing Technologies* (2017).

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

[8] Datta, T., Apthorpe, N., and Feamster, N. Developer-friendly library for smart home iot privacy-preserving traffic obfuscation. In *Proceedings of the 2018 Workshop on IoT Security and Privacy - IoT S&P* (2018).

[9] Draper-Gil, G., Lashkari, A. H., Mamun, M. S., and Ghorbani, A. A. Characterization of encrypted and vpn traffic using time-related features. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy* (2016).

[10] Dwork, C. Differential privacy: A survey of results. In *Encyclopedia of Cryptography and Security, 338-340.* (2011).

[11] Dwork, C., Kenthapadi, K., Mcsherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology* (2006).

[12] Dwork, C., and Lei, J. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposiom Theory Computation* (2009).

[13] Dwork, C., and Roth, A. *The algorithmic foundations of differential privacy.* Boston: Now., DOI: 10.1561/0400000042, (2014).

[14] 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 *Proceedings of 2012 IEEE Symposium on Security and Privacy* (2012).

[15] Erlingsson, U., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security (pp. 1054-1067). ACM.* (2014).

[16] Garrett, T., Dustdar, S., Bona, L. C., and Duarte, E. P. Traffic differentiation on internet of things. In *Proceedings of 2018 IEEE Symposium on (pp. 142-151). In Service-Oriented System Engineering (SOSE)* (2018).

[17] Hamza, A., Gharakheili, H. H., Benson, T. A., and Sivaraman, V. Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity. In *In Proceedings of the 2019 ACM Symposium on SDN Research* (2019).

[18] He, J., and Cai, L. Differential private noise adding mechanism and its application on consensus. In *arXiv:1611.08936v2* (2017).

[19] Herrmann, D., Wendolsky, R., and Federrath, H. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifie. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security - CCSW 09* (2009).

[20] Kawai, H., Ata, S., Nakamura, N., and Oka, I. Identification of communication devices from analysis of traffic patterns. In *13th International Conference on Network and Service Management (CNSM)* (2017).

[21] Kumar, D., Shen, K., Case, B., Garg, D., Alperovich, G., Kuznetsov, D., and Durumeric, Z. All things considered: an analysis of iot devices on home networks. In *In 28th USENIX Security Symposium* (2019).

[22] Liu, J., Zhang, C., and Fang, Y. Epic: A differential privacy framework to defend smart homes against internet traffic analysis. *IEEE Internet of Things 5(2), 1206-1217* (2018).

[23] Makhoul, J. A fast cosine transform in one and two dimensions. In *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1980).

[24] Marchal, S., Miettinen, M., Nguyen, T. D., Sadeghi, A. R., and Asokan, N. Audi: Toward autonomous iot device-type identification using periodic communication. *IEEE Journal on Selected Areas in Communications 5(2), 1206-1217* (2019).

[25] McKinney, W. pandas: a foundational python library for data analysis and statistics. python for high performance and scientific computing. In *Proceedings of 35th IEEE Symposium onSecurity and Privacy Workshops (SPW)* (2011).

[26] Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O., and Elovici, Y. Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the Symposium on Applied Computing - SAC 17* (2017).

[27] 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 *In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017).

[28] Sivanathan, A., Sherratt, D., Gharakheili, H. H., Radford, A., Wijenayake, C., Vishwanath, A., and Sivaraman, V. Characterizing and classifying iot traffic in smart cities and campuses. In *Proceedings of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2017).

[29] Wang, T., and Goldberg, I. On realistically attacking tor with website fingerprinting. In *Proceedings on Privacy Enhancing Technologies* (2016).

[30] Wang, W., Zhu, M., Wang, J., Zeng, X., and Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *IEEE International Conference on Intelligence and Security Informatics (ISI)* (2017).

[31] Zhang, W., Meng, Y., Liu, Y., Zhang, X., Zhang, Y., and Zhu, H. Homonit: Monitoring smart home apps from encrypted traffic. In *In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018).

[32] Differential Privacy Overview - Apple. https://tools.ietf.org/html/rfc2460.

[33] TCP Dump. http://www.tcpdump.org/manpages/tcpdump.1.html.

[34] Wireshark. https://www.wireshark.org/cacetech.html.

[35] Internet Protocol, Version 6 (IPv6) Specification, 1998. https://tools.ietf.org/html/rfc2460.

[36] There is No Place Like [A Connected] Home, 2017. https://www.mckinsey.com/spContent/connected$_h$omes/index.html.

[37] Net neutrality dies on June 11th, 2018. https://www.theverge.com/2018/5/10/17338978/net-neutrality-end-date-fcc.

[38] The 14 biggest announcements from Amazon's surprise hardware event, 2018. https://www.theverge.com/2018/9/20/17883242/amazon-alexa-event-2018-news-recap-echo-auto-dot-sub-link-auto-microwave.

[39] The Connected Home Market, 2018. https://www.mckinsey.com/spcontent/connected$_h$omes/pdf/mckinsey$_c$onnectedhome.pdf.

[40] The General Data Protection Regulation, 2018. https://eugdpr.org/.