Data Instrumentation from IoT Network Traffic as Support for Security Management

Andressa Vergütz, **Bruna V. dos Santos**, **Burak Kantarci** and **Michele Nogueira**

***Abstract*—The Internet of Things revolutionizes human life by inaugurating scenarios such as smart homes. However, IoT devices contain much sensitive information about users and devices from a security and privacy perspective. Through traffic-based attacks, adversaries map changes in traffic rates to a particular in-home user's actions. An efficient IoT data instrumentation may protect users against traffic-based attacks by giving valuable information to adaptive network security solutions. Nonetheless, no work performs data instrumentation or uses feature exploration to reveal relevant features to assist network security management. Thus, this article introduces IoTReGuard, an IoT Method to Reveal and Guard IoT Network Traffic Features. IoTReguard aims to explore network traffic features to reveal the most relevant ones and hide them to protect users' privacy. By IoT network feature exploration and data instrumentation, IoTReGuard provides valuable information on network traffic features to mask critical features. Results showed that IoTReGuard reduced from 70% to 20% of F1-Score on identifying the IoT devices, improving user privacy.**

***Index Terms*—Security management, IoT, smart home, network data instrumentation, privacy.**

I. Introduction

THE advances of next-generation wireless technologies, such as the Fifth Generation (5G) cellular network, have boosted the advance of the Internet of Things (IoT) [1]. IoT collects, quantifies, and provides insights about the surrounding environment [2]. It revolutionizes human life by inaugurating the concept of smart scenarios such as smart home, smart hospital, smart vehicles, among others [3]. In the real world, there are situations that mix devices from different smart scenarios due to user needs. For instance, a smart home can contain typical IoT devices for home monitoring and health-related devices to monitor the resident's health. Thus, this work considers a smart home with various IoT devices. Forecasts suggest that, by 2025, there will be more than 75 billion IoT devices in use, projecting to reach a massive total of 79.4 zettabytes of data volume [4].

Manuscript received April XXX, 2022; revised July XXXX, 2022 and December XXX, 2022. This work was supported in part by the National Council for Scientific and Technological Development (CNPq/Brazil), under grants #432204/2018-0 and 313844/2020-8; supported by FAPESP/Brazil, under grants #2018/23098-0 and 2021/06733-6; and CAPES/Brazil.

A. Vergütz is with the Computer Science Department at Federal University of Paraná, Curitiba, Brazil (email: avergutz@inf.ufpr.br). Bruna V. dos Santos is with Dept. of Information Technology at Federal University of Santa Maria, Frederico Westphalen, Brazil (email: bruna.vitoria@acad.ufsm.br). B. Kantarci is with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: burak.kantarci@uottawa.ca). M. Nogueira is with the Computer Science Department at both Federal University of Paraná and Federal University of Minas Gerais, Belo Horizonte 31270-901, Brazil (email: michele@dcc.ufmg.br)

IoT devices contain sensitive information about users and devices from a security and privacy perspective. Network traffic analysis poses major concerns regarding data privacy and security [4]. An adversary inside or near a smart home may exploit these devices' innate wireless medium to capture sensitive information about the user and his/her activities [5]. This malicious behavior is known as traffic-based attacks or traffic-based side-channel attacks [6]. For example, a motion sensor allows a user to detect any movement in a physical space, but the sensor has only two states (motion and no-motion). If an adversary reveals the current state of the sensor, the adversary shows the user's presence at home. Hence, it is necessary to deal with unstructured and unorganized network traffic data collected from the IoT environment to assist defenses in avoiding such attacks. IoT data support smart data-driven decisions to improve security and privacy.

Efficient IoT network data analysis provides data knowledge for security [7], [8]. However, this process is only straightforward with an efficient organization and traffic instrumentation. Two actions are essential to gain actionable and valuable insights for network security: network data instrumentation and feature exploration. Meaningful network feature is helpful information since it reveals the traffic feature representing an IoT device behavior. The most relevant features, once discovered, allow the design of efficient defense mechanisms hiding key network traffic features to avoid traffic-based attacks.

Nonetheless, no works today perform data instrumentation or use feature exploration to reveal relevant features to employ in network security management. There are works employing feature exploration and ML algorithms to removal of ambiguous traffic [9], to identify applications [10], [11], [12], IoT devices [13], malicious activities [14], [15]. Other works focused on protecting against traffic-based attacks by following masking approaches like dummy traffic generation [16], [17], [18], [5], and packet padding [19], [20], [21]; however, ignoring IoT feature exploration. An efficient IoT data instrumentation may protect users against traffic-based attacks, boosting the users' adoption of smart scenarios.

This article introduces **IoTReGuard**, an **IoT** Method to **Re**veal and **Guard** IoT Network Traffic Features. IoTReguard method aims to explore network traffic features to reveal the key ones and mask them to protect users' privacy. It innovates by quantifying the potential impact of feature exploration when exploited by adversaries on IoT network traffic. When the network traffic is intercepted and all features making up the traffic are fed into feature exploration techniques, adversaries can discover the most relevant features of the

traffic patterns to represent the captured data. Therefore, IoTReGuard reveals the key network traffic features through a feature exploration employing statistical techniques. Then, it masks network traffic data or specific features by obfuscation techniques to prevent traffic-based side-channel attacks. The major contributions of this article are as follows.

- IoTReGuard, a method to reveal the key network traffic features and mask smart home devices. It quantifies the potential impact of feature exploration when exploited by adversaries on IoT network traffic.
- An efficient IoT network feature exploration and data instrumentation to provide valuable information for network security management. The IoT data instrumentation assists efficiently in decision making and knowledge acquisition for adaptive defense mechanisms.

The IoTReGuard method evaluation follows a trace-driven approach under the *IoT Traffic* dataset [22]. The dataset contains network traffic from 30 IoT and non-IoT devices over 20 days. The IoTReGuard evaluation has two main goals: *(1)* analyze the potential impacts caused by adversaries through feature exploration (*RevealFeat*), *(2)* analyze and compare quantitatively different obfuscation techniques against traffic-based attacks *GuardFeat*. Particularly, *RevealFeat* seeks to analyze the network traffic features through IoT data instrumentation and feature exploration. While, *GuardFeat* aims at evaluating the effectiveness of protections against traffic-based attacks. We compare the IoTReGuard defense technique based on fake traffic generation with the packet padding technique from [19]. The method reduced from 70% to 20% of F1-Score on identifying the IoT devices in the masking traffic results.

This article proceeds as follows. Section II discuss the related works. Section III introduces the IoTReGuard method. Section IV presents the performance evaluation methodology and discusses the results. Finally, Section VI concludes the article and discusses future works.

II. Related Works

IoT network traffic analysis through feature exploration and ML algorithms has been widely investigated with the aiming of improving the network performance [23], [24], the removal of ambiguous traffic [9], defending the network against attacks [25], and the characterization of IoT devices behavior [22]. However, only some works perform data instrumentation or use feature exploration to reveal relevant features to employ in network security management. By analyzing the most relevant features, adversaries can exploit them to infer information that makes attacks more efficient [25]. The main challenge is identifying the relevant features from network traffic suitable for each network context. Thus, the following subsections present the main works applying feature exploration for different purposes and the primary defenses against traffic-based attacks.

A. Network Feature Exploration and Instrumentation

Most studies employ feature exploration and selection to reduce data dimensions, abandon irrelevant information, and improve classification [26]. For instance, Principal Component Analysis (PCA) extracts the most relevant information from redundant or noisy data. It offers detailed information on the sample representation of data acquired from the network. It creates new variables, called Principal Components (PCs), through a linear combination of the original features. As PCs may lead to entry points for adversaries in IoT networks, these methods lead to the development of defenses against passive attacks. The authors in [27] proposed a new PCA-based method for intrusion detection systems, and the authors in [28] used PCA for IoT network traffic anomaly detection since it reduces dataset complexity. However, only some works use feature selection to reveal relevant features to employ in security management.

Other works used network traffic analysis to identify applications [10], [11], [12], IoT devices [13], malicious activities [14], [15], or information exposure by IoT devices [29]. The authors in [12] compared different feature extraction techniques in network security context. However, they considered string features, such as the host website name, and did not consider the exploration of relevant features. The authors in [13] identified IoT and non-IoT devices by ML classifiers, while the authors in [14] used network traffic features and ML algorithms to detect malware. However, the works should have considered feature exploration and selection. The authors in [29] showed that the signatures are unique among hundreds of network traffic packets from IoT devices, which increases the attacker's capabilities. The authors in [15] proposed a device identification system to detect malicious activities in smart environments. They applied a correlation-based feature selection algorithm to identify and remove any redundant features. Although they benefit from feature selection to improve the classification task, the authors did not analyze the feature impact on traffic-based attacks, abstaining from such attacks.

Network traffic analysis and feature exploration may assist the development of defense mechanisms tailored to IoT [30]. However, there are no works using feature exploration for this purpose. The authors in [31] discovered general security and privacy with smart home devices, including insecure TLS implementation, unencrypted traffic, and pervasive use of the tracking and advertising services. However, no defense mechanism was applied to protect in-home users' privacy. Based on the literature, there are few works exploring network traffic features. Once IoT data is complex, unstructured, and keeps increasing [30], there are research opportunities to examine network traffic features to give insights into security management, especially when considering IoT users' privacy. Although several works in the literature proposed methods and architectures to address specific problems, such as traffic identification [22], [32], [15], big data analysis [33], [34], [35], and ML application on network traffic [36], [37], [26], none network traffic analysis considering feature exploration and data instrumentation was proposed to assist adaptive network security solutions.

B. Defense Techniques Against Traffic-based Attacks

The ability to infer information about encrypted wireless communications via side-channels data (*e.g.*, packet and flow-

level features) has been previously established [38], [39]. Given the limited-purpose nature of many IoT devices (*e.g.*, a smart power switch only function is turning on or off), it is often straightforward for an adversary to map changes in traffic rates to a particular user's actions. There are different masking approaches to prevent adversaries from exploring network traffic features (or side-channel data leaks). Some of the most popular defenses propose *(i)* to inject dummy (fake) traffic [16], [17], [18], [5], and *(ii)* padding bytes to packets for a fixed or random size [19], [20], [21].

Fake Traffic Injection Approach: The fake traffic injection approach confuses the adversary by adding fake network packets. The authors in [16] proposed an obfuscator that sends fixed-length packets at a fixed interval. However, different applications, scenarios, and environments will present different behaviors from the feature perspective. Thus, it is essential to analyze and explore features to understand the particularities of the environment. The authors in [40] added fake network traffic in the transmission sequence randomly based on logistic regression. Nonetheless, the authors did not discuss different network traffic features that adversaries can use.

The authors in [5] proposed a mitigation approach based on traffic generation to hide the device states. The authors in [41] proposed an IoT traffic shaping considering fake traffic generation, and the authors in [42] proposed a defense system to reduce the private information leaked through IoT device network traffic. Moreover, they analyzed the principle features from network traffic traces by feature selection algorithms (*e.g.*, PCA). From 10 features, PCA indicated to use 8 features. However, the feature analysis did not consider different network traffic features (*e.g.*, timestamp, and a number of bytes), only considering the statistical proprieties of traffic rate (*e.g.*, mean and max). It is possible to perform side-channel attacks based on different features.

Packet Padding Approach: The packet padding employ several padding strategies to reduce the accuracy of classifiers. Maximum transmission unit (MTU) padding, for example, is a well-known mutation technique that consists of padding all the packets to the maximum payload size MTU (*e.g.*, 1500 bytes) [16]. Random padding is another technique that consists of randomly padding the packets, and Random 255 inserts a random value between 1 and 255 bytes [43]. Linear padding increases the packet size to its nearest 128 multiples, while exponential padding increments the packet to the nearest power of two [19]. Mouse/elephants padding increases the packet sizes to 100 or 1500 bytes accordingly with a threshold. Values less than 100 are incremented to 100 bytes, while sizes greater than 1000 are increased to 1500 bytes. However, packet padding can result in a drastic overhead in the amount of data sent, causing performance degradation related to delay and bandwidth consumption [16]. Furthermore, some works proved that it is still possible to classify the traffic, even when implementing these techniques [44].

Other padding techniques aim to confuse the classifier into classifying the target application traffic as another type. The authors in [20], [21] proposed a system to mobile fingerprint applications and mask the packet size adding random sizes in the flow, following a packet padding approach. However, these works did not explore the feature importance considering the application. The authors in [16] showed that simply padding or fragmenting packet lengths to a constant size is not sufficient for hiding user activity since using other features (*e.g.*, total bandwidth) still achieves high performance (98% of accuracy) on traffic-based attacks. The authors in [19] proposed a dynamic packet padding technique to mask IoT network traffic. The authors created padding levels. Each level compares the packet number of bytes to increment the length, *i.e.*, the level numbers denote the minimum length of the packets that will belong to the respective level after padding. However, they did not perform any feature exploration. The decision on using packet size was based on other works that used the same network traffic feature. Thus, there are none works using the feature exploration to assist the adaptive defense against traffic-based attacks.

III. The IoTReGuard Method

This section introduces the IoTReGuard method, which reveals and masks network traffic features on smart homes. The IoTReGuard method organizes and structures network traffic data. IoTReGuard is based on two main steps that support each other: *(i) reveal features* and *(ii) guard features*. Different from the literature, IoTReGuard reveals the key network traffic features through an exploration employing statistical techniques. Based on the revealed features, IoTReGuard masks network traffic data by obfuscation techniques to prevent traffic-based attacks. It preserves in-home users' privacy without harming the network communication since the method allow to mask only the key network traffic features. The following subsections describe the attack overview and IoTReGuard's main steps.

A. Traffic-based Side-Channel Attacks Setting Overview

Fig. 1 shows a setting overview of a passive network traffic-based attack on a smart home, in which an adversary aims to infer user activities. We consider an adversary located physically within the wireless range of the targeted users' devices [5], [19]. Hence, the adversary can install the sniffer in the gateway only once and manage it remotely. Alternatively, it could compromise a device inside the smart home remotely and turn it into a sniffer. In this way, the adversary may never need to be present. In all these cases, the adversary can eavesdrop on wireless network communications transmitted by the devices. These devices include cameras, motion sensors, temperature sensors, switches and triggers, lighting sensors, smartwatches, laptops, and others. As a typical smart home setting, they transmit network traffic to the AP (access point), the AP transmits to the router and the router to the Internet. The adversary only needs to sniff the network traffic without interrupting passively. Therefore, the attacker may stay active long enough without being detected by the victim.

The adversary sniffs network traffic side-channel information such as packet size, the interval between packets, and others [19]. Thus, the traditional means of encrypting packets are no longer feasible approaches from the privacy perspective since it only encrypts the packet payload. Using side-channel

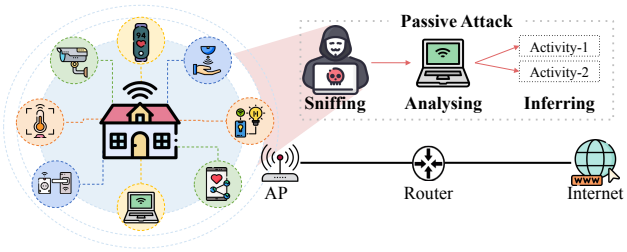
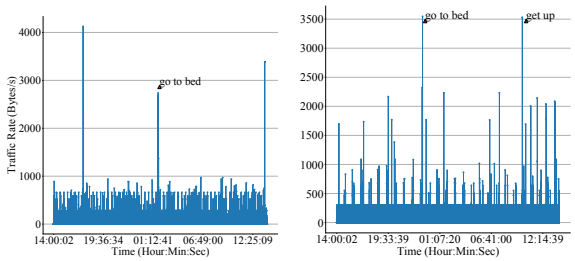


Fig. 1. Setting Overview on Traffic-based Side-Channel Attacks

information, the adversary can build classifiers and infer user behavior. For instance, the packet size and timestamp in ML techniques enables inferring if an individual goes to bed, health conditions, assistant interactions, and others [42]. Fig. 2 presents the network traffic rate trace (in bytes/s) of 2 IoT devices over 24 hours (*hour:min:sec* format following the Brasília Standard Time) from a smart home. Figs. 2(a) and 2(b) display significant peaks from the network traffic rate of a motion and sleep sensor, respectively. The relation between the time of each traffic rate peak from the sensors may indicate users go to bed around 01:05 a.m., and get up around 12:15 a.m. Thus, by analyzing these devices, the adversary can deduce the user went to bed late at night and woke up near noon, learning his/her daily routine.



(a) Belkin Wemo Motion Sensor (b) Withings Smart Sleep Sensor

Fig. 2. User Interactions over 24 hours increase IoT Devices Traffic Rate

B. IoTReGuard Setting Overview

IoTReGuard method reveals and masks the key network traffic features on representing the network traffic behavior and inferring user activities. Hence, IoTReGuard method works based on side-channel features from the segment header. Fig. 3 shows the system structure of IoTReGuard. It can be deployed either on a home router or home AP that serves as a gateway for the home network. From the AP, IoTReGuard sniffs the network traffic to reveal key network traffic features. In this step, the adversary can still sniff the network traffic. Then, IoTReGuard takes additional steps to further obscure network traffic and user activities in-home privacy. It masks network traffic through obfuscation techniques such as the packet padding approach and injecting fake traffic. Therefore, the masking step of IoTReGuard makes unfeasible the adversary success on collecting useful features since the network traffic feature is masked. Thus, IoTReGuard identifies different IoT devices by side-channel information to reveal devices and users' behaviors and then masks this information to prevent passive attacks.

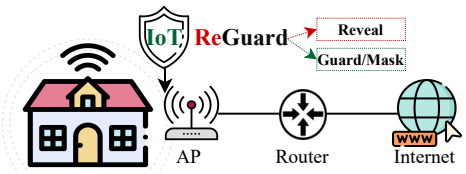


Fig. 3. IoTReGuard Position in the Smart Home

C. Revealing Network Traffic Features

This section presents the *reveal features* step of IoTReGuard, *i.e.*, the feature exploration under the perspective of adversaries. Different from the literature, IoTReGuard highlights the possible damages of a thorough network traffic analysis by adversaries. For the sake of completeness, Fig. 4 shows an ML-based network traffic exploration and classification overview that involves five steps: data collection, feature pre-processing, feature selection, classification, and data-driven decision making. The first three steps compose the feature exploration. IoTReGuard focused on these steps from an adversarial standpoint. For feature exploration, it is necessary to collect network traffic data; hence, this work considers a smart home, containing various devices such as a camera, smartwatch, and ambient sensor that generate traffic. Hence, data input for feature exploration is network traffic containing data packets generated in an IoT setting. The next subsections detail the feature exploration steps.

1) Feature Pre-Processing from Network Traffic: To efficiently find out the meaningful features for traffic classification, it is essential to pre-process the collected network traffic through data cleaning and extraction. Note that meaningful features are considered as critical features since the attackers may leverage for in-home users data inference. Cleaning the traffic capture by removing unnecessary traffic data and extracting network features is paramount. Feature extraction involves information from the segment header, such as segment time. Effective feature extraction has to enclose the output features as much as possible the information available from the collected network traffic. It is possible to extract features at the packet or flow level. Flow-level data is based on a 5-tuple structure: source/destination IP addresses, source/destination port numbers, and protocol (TCP/UDP). Analyzing the 5-tuple, flow-level data contains all network packets that belong to the same flow to extract features such as data volume per flow, number of packets per flow, and others. In contrast, packet-level considers features related to individual network packets, *e.g.*, packet size, timestamps, and others. Moreover, statistical primitives (*e.g.*, min, max, mean, and variance) computed from the features can increase data granularity and assist in device traffic classification.

Feature pre-processing output follows a time-series or packet-series format for flow-level and packet-level, respectively. For instance, the output of flow-level feature pre-processing consists of a time series containing packet size, packet time, and other traffic features. As traffic is generated by multiple devices, MAC address serves as a label to point

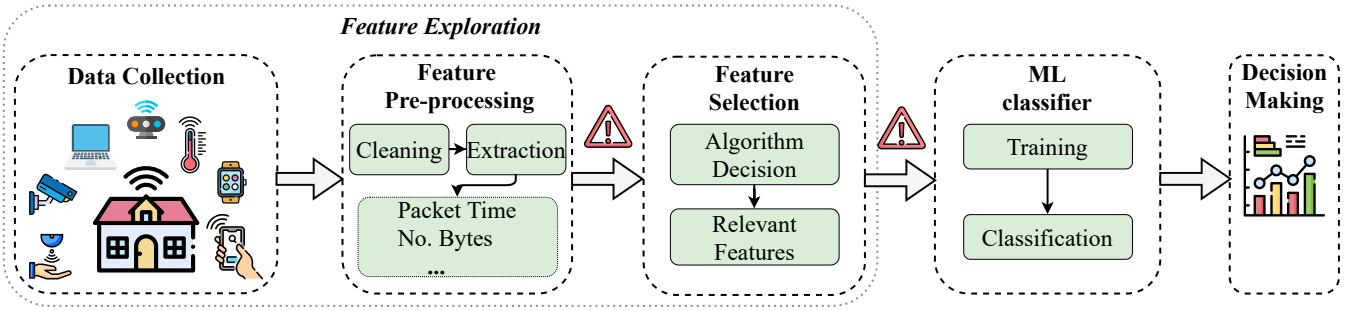


Fig. 4. ML-based Network Traffic Classification Overview

out the association between traffic and a device. Through the output of feature pre-processing, it is possible to achieve network traffic feature information, which can be used by an adversary. Although the packet payload undergoes a cryptography process, network traffic features are extracted from the segment header. In general, the content of the payload is not deeply analyzed. Thus, even in the case of encryption, features can be extracted. Hence, adversaries can infer the user information and lead to security implications by only analyzing the network features (*i.e.*, perform network traffic-based attacks). Inferring user information have consequences such as learning users daily routines. Therefore, network features, when statistically align with human behavior patterns, may lead to privacy violations.

2) Feature Selection from Network Features: After feature pre-processing and extraction, it is necessary to analyze, select, and combine meaningful features. The main steps to discover the most relevant network traffic features are: 1) the removal of features that are less significant, and 2) feature contribution analysis. It is necessary to normalize, discretize the features, and remove noise. The removal of features that are less significant can be achieved by using the variance of features. Low variance results in less significance of a feature to contribute to the characterization of the network traffic [45]. Feature contribution indicates whether/how a particular feature plays a role in traffic classification. Thus, the relevant feature set is obtained by eliminating the features with low variance, as well as those redundant and less relevant.

Feature selection algorithms identify the most relevant features. For instance, PCA is a linear dimensionality reduction technique [46] that seeks to map data points from high dimensional space to a low dimensional space while keeping all the relevant linear structures intact. Hence, through PCA, an adversary can discover the key features that most represent the original network traffic. In other words, they assist adversaries in learning traffic behavior and performing traffic-based attacks and even zero-day attacks. Without this knowledge, the adversary performs an attack less efficiently since it uses several network traffic features. Thus, based on feature selection, the adversary learns the network traffic patterns and discovers the most significant feature, assisting in attack modeling. The adversary can achieve higher damage through the attack, improving the impact and reducing the complexity. For instance, assuming the most relevant feature is related to the network packet time, the adversary would perform the attack using mostly timing features since they represent better the network traffic behavior. Such attacks may jeopardize user data privacy and reveal sensitive information.

Although such key features assist malicious adversaries to perform attacks, they also guide the design of security and privacy solutions by obfuscating relevant features and making unfeasible the discovery by adversaries. For instance, instead of masking all network traffic data or specific features without knowledge about relevance, thorough feature exploration can mask only the most representative feature reducing time implementation and guaranteeing the obfuscation of essential and representative features. Moreover, masking only key network traffic features may reduce network overhead because it is necessary to inject a smaller amount of false traffic than when masking all network traffic features. Therefore, by feature exploration of IoTReGuard is possible to make a data-driven decision on masking features.

D. Masking Network Traffic Features

The second step of IoTReGuard masks network traffic features to avoid traffic-based attacks and user activities inferring. IoTReGuard method employs obfuscation techniques such as the padding approach and injecting false traffic. Note that IoTReGuard is dynamic since it supports different obfuscation techniques. The goal is to mask side-channel information in the best way possible. It is not in the scope of this work propose a new obfuscation technique. There are several obfuscation techniques in the literature [42], [25], [47]; however, there are still open issues regarding these techniques such as the trade-off between privacy and overhead. Thus, IoTReGuard follows a dynamic approach making it feasible to employ different techniques. Moreover, different from the literature, based on the first step of feature exploration from IoTReGuard, it is possible to mask only the relevant network traffic features.

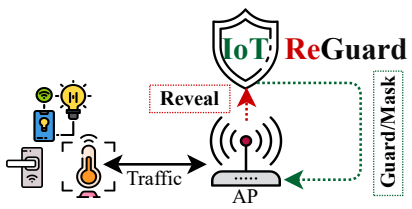


Fig. 5. IoTReGuard Steps

In general, obfuscation techniques mask side-channel information by injecting or altering traffic data [19], [42]. For

instance, following the injection fake traffic approach, IoTReGuard generates network traffic based on the MAC address of the original IoT devices identified in the first step, feature exploration. In this way, the fake network traffic contains similar patterns to the original traffic since it considers the device type. Thus, it is difficult to distinguish the injected fake traffic from the real demands due to their behavior similarities. Moreover, network traffic injection usually follows the number of devices connected to the smart home or the original traffic size. In other words, it is possible to inject from 10% to 100% of fake traffic and shuffle with the original data. To analyze the masking performance, IoTReGuard compares classifiers performance results on identifying IoT devices. Therefore, first, it identifies devices only considering the real network traffic. Then, it adds the fake traffic and identifies the devices by classifiers. If the performance of the classifiers decreases, it means good masking and privacy protection performance.

IV. Performance Evaluation

The performance evaluation of the IoTReGuard method follows a trace-driven approach. It has two main goals (*i.e.*, it follows the two main steps of IoTReGuard): *(1)* analyze the potential impacts caused by adversaries through feature exploration (*RevealFeat*), *(2)* analyze and compare quantitatively different obfuscation techniques against traffic-based side-channel attacks (*GuardFeat*). *RevealFeat* seeks to analyze the network traffic features through feature exploration in an adversarial perspective. It lies in highlighting the key network traffic features that can be used by adversaries on traffic-based attacks considering the application of feature exploration. *GuardFeat* aims to evaluate the effectiveness of common protections against traffic-based attacks. In this work, we compare the IoTReGuard obfuscation based on fake traffic injection (called FAKE-A) with packet padding (named PADD-A). Thus, next sections present the evaluation methodology, obfuscation techniques, and discuss the results in an adversarial perspective.

Dataset Characteristic: The case study follows a trace-driven approach considering the *IoT Traffic* dataset [22] that presents network traffic from 20 days, containing 20 packet capture (PCAP) files. Each one corresponds to 24h of the capture of the traffic generated by 30 IoT and non-IoT devices. The dataset contains hubs, healthcare devices, monitoring cameras, switches and triggers, light bulbs, air quality sensors, electronics, and several non-IoT devices (*e.g.*, laptops). The dataset serve as input for *RevealFeat* and *GuardFeat* analysis. Thus, we extracted network traffic features (*e.g.*, packet size, timestamp, and the number of packets) and the associated capture time. We filtered the devices by the MAC address and created the device label. Such labels assist the feature exploration and serve as input for the supervised ML-algorithms.

Details of Feature Exploration: To efficiently highlight the essential features in *RevealFeat* and use them in *GuardFeat*, we extract network features from the segment header by Tshark. The packet payload is not included in the feature extraction. Thus, the input file for the analysis is composed of network features and the associated capture time. The capture time encompass the trace capture day from the dataset. Based on this, we normalize the capture time to an interval from 1 to 20 days. We extract network features following two evaluation scenarios: *(i)* the packet scenario (PS), where we employ the features related to the network packet size; and *(ii)* the flow scenario (FS), that we use the features related to the flow, such as flow duration, as shown in Table I. A flow is the data-plane of packets between sender and receiver that shares key IP header information (*i.e.*, 5-tuple information). Flow duration is the sum of all packets timestamp.

TABLE I
Network and Statistical Features considered in the Scenarios

	Ntw. Traffic Features	Statistical Features
PS	Packet Size, Timestamp, Inter-packet-time	Mean, standard deviation, min, max, sum, median
FS	No.Packets, Packet Size, End and Start Timestamp, Inter-packet-time, Flow Duration	Mean, standard deviation, min, max, sum, median

Precisely, from the 20 pcaps, in the PS scenario we extract the packet size, timestamp, and inter-packet-time (IAT), while in the FS scenario we extract the number of packets, flow size, start-end timestamps, inter-packet-time, and flow duration. In this work, the IAT consists in the time between two sequential network packets. The start timestamp consists of the time the first packet was transmitted, and the end timestamp corresponds to the time of the last packet was transmitted, both packets belonging to the same flow. Furthermore, in data cleaning, the analysis ignores packets destined to DNS servers, broadcast, and packets with size equal to 0 since they are packets without relevant content. The feature extraction and cleaning output consists in 20 CSVs files for each capture day containing the packets network features, and the device label. Note that categorical features, such as IP and MAC address, are only used to analyse the traffic behavior in the *RevealFeat*.

After extracted the network traffic features by Tshark, from the CSVs, we compute statistical features to increase the dataset details, such as *mean, standard deviation, min, max, sum*, and *median*. In the PS scenario to calculate the values for the statistical features, we group the packets belonging to the same device and create small samples of the network traffic features (samples size = 3). The samples have size 3 because the device with smallest amount of traffic was BlipcareBP (with 57 packets). Hence, we create small samples to not loss information from devices that generate small amount of traffic. Thus, every 3 network packets, we compute the statistical features and label according to the device MAC address. For the FS scenario, we compute the statistical features from the packets belonging to the same flow. After compute statistical features, each device has its own network and statistical features for the PS and FS scenarios. Then, in each scenario (PS and FS), we merge all devices features, including the device label, into one unique final file. This final file serve as input for both analysis, *RevealFeat* and *GuardFeat*. In *RevealFeat* the final file is used as input for the feature selection algorithms, and in *GuardFeat* serve as input for the obfuscation techniques and classifiers.

Feature Selection Algorithms: The *RevealFeat* considers two well-known statistical methods: FA and PCA since the former aims to reveal the latent features whereas the other assumes correlation between features. Network traffic features often involves a high dimension and correlated data since it includes many data. As PCA and FA are also considered as dimension reduction techniques, they are suitable for network traffic analysis [46]. Therefore, in *RevealFeat*, PCA and FA follow two phases. PCA: 1) pre-processes data to normalize the mean and variance, 2) It computes co-variance matrix, eigenvectors, and eigenvalues. Thus, PCA converts the original data into new set of axes called PCs by keeping the most essential features of the original data. FA: 1) normalizes the network data and defines the number of factors, 2) It calculates the correlation matrix to correlate the features. Hence, FA converts the original data into factors containing the correlated features and keeping the most representative features [48]. Feature exploration using PCA and FA follow scripts created in Python, v. 3.8, and R, v. 3.6. In R, we use specific libraries for data analysis, *e.g.*, FactorMiner and FactorExtra.

Machine Learning Algorithms: This work considers supervised algorithms to classify the IoT devices. The dataset contains the MAC address of each device; thus, we assume the adversary has several MAC address to perform the traffic identification. MAC address serves as input to label devices. The *RevealFeat* and *GuardFeat* consider decision tree-based and ensembles supervised algorithms. Precisely, through scikit-learn library, the analyses employ the Classification and Regression Trees (CART) as decision tree-based, the Extreme Gradient Boosting (XGBoost) and Bagging (Bagg) as ensemble algorithms, and Random Forest (RF) as decision tree ensemble. In general, these algorithms work well for classification problems and with unbalanced data [49].

A. Obfuscation Techniques

In this case study, IoTReGuard employed two well-known approaches: injecting fake traffic (FAKE-A) and packet padding (PADD-A). Both obfuscation approaches inject or alter network traffic data belonging to the IoT dataset. Thus, the feature exploration, including cleaning, extraction and preprocessing network traffic features serve as input for the fake traffic injection and packet padding approaches. The fake traffic injection adds fake data into the network capture of the dataset, while the padding approach alters the number of bytes from the devices network traffic capture. Both approaches was developed in Python version 3.8, including libraries, and Scapy Traffic Generator version 2.19. To extract network traffic features from PCAPs files was used Tshark.

Fake Traffic Injection Details (FAKE-A): The most simple and effective method to resist traffic-based side-channel attacks is to add fake traffic onto the transmission sequence to make the adversary unable to distinguish between fake and real network packets [40]. From the MAC addresses and devices labels, IoTReGuard generates fake network traffic data. It creates one fake MAC address for each real MAC address through the Python Generate MAC library (version 1.3). The fake and real MAC addresses are similar because they are used to label devices and create network traffic behavior similar to the original. Hence, as the domestic IoT traffic dataset contains 31 devices, it was created 31 fake devices, totaling 62 devices (reals and fakes). Periodically, when the dataset contains more than 60 fake MAC addresses, we remove them from having a sizeable fake dataset. The fake devices receive random labels to use in the ML algorithms. Moreover, as the case study follows a trace-driven approach, the fake traffic generation follows a constant rate (*i.e.*, sending packets at a fixed frequency) to inject it into the IoT dataset.

Due to the power limitations of IoT devices, the amount of fake data should be added as low as possible. We follow different levels of fake traffic injection to compare the difference amount of false traffic. Table II presents the different levels of amount of fake traffic. 600-FAKE, 1000-FAKE, and 5000-FAKE generate 600, 1000, and 5000 network packets per fake device. In contrast, the Rdm-FAKE adds a dynamic amount of packets ranging from 1000 to 10000 fake packets. We did not change the number of bytes of the fake packets. The Traffic Generator Tool adds a standard packet size for the fake traffic: 83 or 133 bytes. Moreover, the fake packets follow the transport-layer TCP protocol; hence, the Traffic Generator Tool considers Acknowledge (ACK) packets as one of the false packets generated. For instance, in the 600-FAKE level, each fake device generated 600 network packets, including ACKs. The feature extraction groups packets belonging to the same source and destination IP addresses. Therefore, the number of fake packets was reduced to 100, 200, and 1000 fake packets per device, respectively, for the 600-FAKE, 1000-FAKE, and 5000-FAKE levels.

TABLE II
LEVELS CONSIDERED IN THE FALSE TRAFFIC INJECTION

Levels	Amount of False Packets	Total Bytes
600-FAKE	600 false packets p/ device	1.7 MB
1000-FAKE	1000 false packets p/ device	2.8 MB
5000-FAKE	5000 false packets p/ device	14 MB
Rdm-FAKE	1000-10000 false packets p/ device	135 MB

The entire IoT dataset contains 11.3 GB of network traffic data. Table II shows the total amount of bytes injected by each fake traffic generation level, not been a massive amount of fake traffic when compared to the amount of real network traffic. Moreover, in the network traffic generation, we create dynamic IP addresses for all fake devices. The devices communicate with the smart home gateway. Hence, the destination IP address receives the same IP address from the original gateway. In this way, the entire network traffic, including original and fake network traffic, communicates with the smart home gateway, following a typical IoT environment. We analyze all FAKE-A levels of amount of traffic in both FS and PS scenarios. Thus, we evaluate each FAKE-A level by injecting the fake MAC addresses, fake labels, and fake network packets into the original IoT dataset. In other words, the fake traffic is mixed with the original traffic.

Packet Padding Details (PADD-A): We compare the FAKE-A approach with the packet padding approach (PADD-A) proposed by [19]. The authors proposed a four-level padding

strategy for IoT named 100-PADD, 500-PADD, 700-PADD, and 900-PADD, where the numbers denote the minimum length of the packets that will belong to the respective level after padding. The length corresponds to the number of bytes (*i.e.*, packet size). The lower the level, the fewer the number of extra bytes inserted into the packets. In the lowest level (100-PADD), the lengths of the packets are kept close to the original values. As the padding level increases, the privacy and overhead also increase. The authors have considered the trade-off between privacy and overhead. When the network is overloaded, the level 100-PADD performs better because it generates less overhead. Otherwise, when the network is idle, it is more appropriate to implement level 900-PADD, which produces the best privacy improvements. We compare and test all levels of PADD-A to analyze the performance. As [19], we compare FAKE-A and PADD-A with well-established packet padding approaches, including: linear, exponential, mice/elephants, random 255, random, and MTU.

V. Results

This section presents the performance evaluation results of IoTReGuard. The following subsections present the revealing and masking of relevant network traffic features in a smart home scenario.

A. RevealFeat: Feature Revealing Results

In the feature exploration (*RevealFeat*), we used the PCA and FA to highlight the most relevant network features from the IoT dataset. The number of PCs and Factors is less than or equal to the number of original features. As we follow two evaluation scenarios, PS and FS, in the packet scenario we consider the packet size (number of bytes), timestamp, number of packets, and IAT. In contrast, in the flow scenario we consider the number of packets, packet size (number of bytes), end and start timestamp, IAT, and flow duration. In this work, we use packet size and number of bytes to describe the size of network packets. For the PS scenario, Table III displays 4 PCs and 3 Factors results since this scenario contains four network traffic features. The number of factors is only three because the first three already contain the most representative features. The PC1 heavily weights features related to the size of packets, achieving 0.70 of correlation between the number of packets and number of bytes. PC2 heavily weights features related to the time, with a high correlation (0.71) between timestamp and IAT. The proportion of variance and cumulative proportion under PCA assist in understanding the most representative PCs (and features). The proportion of variance analyzes how much each Factor or PC contains variance representing the original data. Closer to 100%, better to represent the data. The cumulative proportion quantifies the representation value. PC1, PC2, and PC3 represent 50%, 32%, and 17% of the original data, respectively, achieving 99% of cumulative proportion.

FA analysis presents similar results. Table III presents the FA results. F1 heavily weights features related to the size and number of packets (99% of correlation), F2 strongly correlates with time-related features (60% and 23% of correlation). F3 presents a high description for IAT. F1, F2, and F3 represent 71%, 15%, and 14% of the original network data, respectively, achieving 100% of cumulative proportion. Therefore, both analyses, FA and PCA, in PS presented the features related to the size of packets contribute more to representing the original data when compared to the other features.

TABLE III
PS: PCs and Factors Results from Network Traffic Features)

Features	PCA			FA		
	PC1	PC2	PC3	F1	F2	F3
No.Packets	**0.70**	-0.03	0.04	**0.99**	0.11	-0.07
No.Bytes	**0.70**	-0.04	0.10	**0.99**	0.01	0.00
Timestamp	0.10	**0.69**	-0.71	0.06	**0.60**	0.25
IAT	0.03	**0.71**	**0.70**	-0.04	**0.23**	0.57
Var. Prop.	0.50	0.32	0.17	0.71	0.15	0.14
Cum. Prop.	0.50	0.82	**0.99**	0.71	0.86	**1.00**

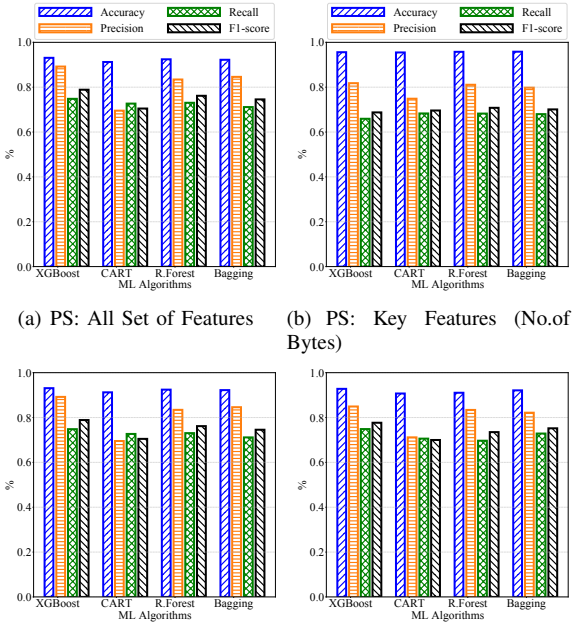
Table IV displays 3 PCs and 3 Factors results for the FS scenario. Although this scenario contains six traffic features, it was unnecessary to create one PC for each feature since they are highly correlated. For instance, PC1 and F1 strongly correlate with start and end timestamp features. The PC2 and F2 heavily weigh features related to the size and number of packets. Finally, PC3 and F3 achieve a high correlation between IAT and flow duration since both features are computed by time information. PC1, PC2, and PC3 achieved 95% of cumulative proportion. F1, F2, and F3 achieved 92% of cumulative proportion. Therefore, the first components and factors have the most representative feature correlation. In the adversary perspective, by analyzing the PCA and FA results, an adversary may decide to use only one feature from each combination since the results showed a high correlation, *e.g.*, the number of packets and start timestamp.

TABLE IV
FS: PCs and Factors Results from Network Traffic Features

Features	PCA			FA		
	PC1	PC2	PC3	F1	F2	F3
No.Packets	0.04	**0.70**	-0.007	0.05	**0.99**	-0.009
No.Bytes	0.04	**0.70**	-0.01	0.05	**0.99**	-0.01
IAT	0.05	0.003	**0.70**	0.07	0.004	**0.93**
Start Time	**0.70**	-0.04	-0.05	**0.99**	-0.05	-0.07
End Time	**0.70**	-0.04	-0.05	**0.99**	-0.05	-0.06
Flow Dur.	0.05	0.009	**0.70**	0.07	0.01	**0.93**
Var. Prop.	0.33	0.33	0.29	0.33	0.33	0.25
Cum. Prop.	0.33	0.66	**0.95**	0.33	0.66	**0.92**

PCA and FA revealed that the most relevant features involve the number of bytes for PS and timestamps, number of bytes, and packets for FS. To confirm features impact in classification, Fig. 6 compares the results for accuracy, precision, recall, and F1-score using all set of features and only the key features for each scenario. XGBoost, CART, Random Forest, and Bagging were used to perform the comparison. Figs. 6(a) and 6(b) show the ML algorithms results using all set of features and only key feature for PS. Using all set of features, the algorithms achieved around 78% of F1-score and more than 80% of precision. Considering only the main key feature,

number of bytes, performance presented a small decrease because the training features reduced from four (timestamp, IAT, number of bytes, and packets) to only one feature. Although decreased performance, it was not so significant when comparing the ML results. Random Forest and Bagging achieved 88% and 92% of precision, respectively, using all features, while using only key features, they presented 81% and 79% of precision, respectively. Therefore, such results confirmed the relevance of key features on describing data.



(a) PS: All Set of Features (b) PS: Key Features (No.of Bytes)

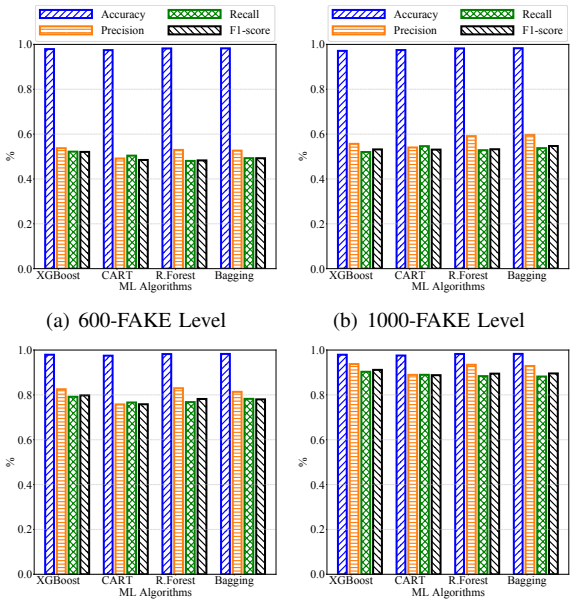
(c) FS: All Set of Features (d) FS: Key Features

Fig. 6. Comparing the use of non-relevant feature by ML Algorithms

Figs. 6(c) and 6(d) show the ML algorithms results using all set of features and only key feature for FS. In both settings for FS, the ML algorithms presented similar results. Regarding all features, the algorithms achieved around 75% and 84% of F1-score and precision, respectively. Considering only the key features, the algorithms presented around 74% and 80% of F1-score and precision, respectively. Hence, the IoT device identification still achieved a high performance only considering the key features revealed by PCA and FA. These results confirm the low relevance of IAT and flow duration presented in the feature exploration results since their use did not affect ML performance. Therefore, feature exploration and ML algorithm comparison indicate that number of bytes and timestamps features are the most relevant features, followed by the number of packets. However, network traffic-based attacks take benefits from this information to learn user routine and violate privacy. These attacks consider packet size (number of bytes) or timestamp as the network traffic feature [25]. It is possible to infer information from a smart home routine by identifying and learning the device's behaviors.

B. GuardFeat: Feature Masking Results

This section presents a quantitative comparison on network traffic feature obfuscation since IoTReGuard supports different obfuscation approaches. IoTReGuard method employed two approaches: injecting fake traffic (FAKE-A) and packet padding (PADD-A). Considering the *PS Scenario* and a hold-out evaluation (testing and training datasets), Fig. 7 displays the obfuscation results related to the fake traffic injection approach (FAKE-A). Without defenses against traffic-based side-channel attacks, classifiers achieved ≈ 82% and ≈ 78% of precision and F1-Score, respectively. Surprisingly, the 600-FAKE level obtained the best results on reducing the performance of the classifiers in the IoT device identification, as shown in Fig. 7(a). As the FAKE-A sends additional traffic to hide real traffic, we expected the heavier the fake traffic, the higher the network traffic obfuscation. However, the obfuscation results showed the contrary. As many traffic samples are given to the supervised classifier, better will be its performance. A rich training dataset with large network traffic samples increases the probability of a classifier identifying the devices with success. Due to this, Rdm-FAKE level presented high performance results ≈ 95% of F1-Score for all classifiers. Rdm-FAKE added from 1000 to 10000 fake network traffic packets per real device (135 MB of fake traffic). Compared to the IoT traffic dataset size (13 GB of real traffic), 135 MB is a small size value. However, it was enough to increase the results of the classifiers.



(a) 600-FAKE Level (b) 1000-FAKE Level

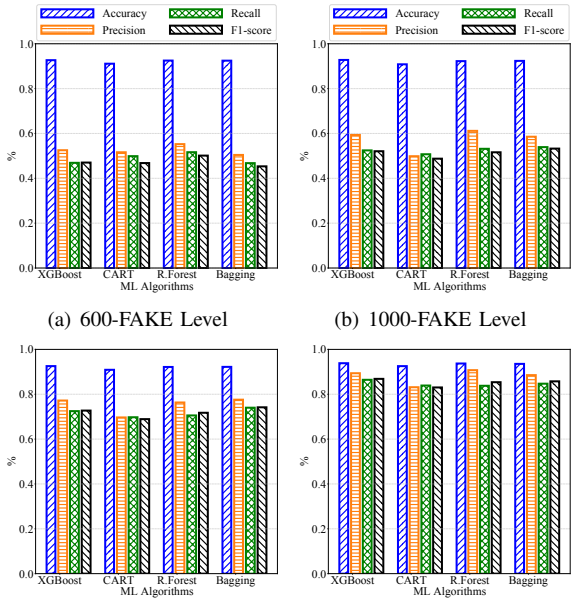
(c) 5000-FAKE Level (d) Rdm-FAKE Level

Fig. 7. PS: FAKE-A Approach Results

The 600-FAKE and 1000-FAKE levels achieved similar results on the network traffic obfuscation due to the amount of fake packets generated. As the FAKE-A follow the TCP protocol, the 600 and 1000 fake packets include ACK network packets. The feature extraction grouped packets belonging to the same source and destination IP addresses. Thus, the number of fake packets was reduced to 100 and 200 for the 600-FAKE, and 1000-FAKE levels, respectively. The amount of fake traffic from both levels justifies the similar results on the classifiers. Nonetheless, it is possible to note a tiny improvement in the 600-FAKE level classifiers results. Therefore, the 600-FAKE presented the best obfuscation results,

which is good for the IoT environment since the less fake traffic added, the lower the network bandwidth overhead. Moreover, most IoT devices generated a small number of traffic packets. Hence, to obfuscate typical IoT devices, a small amount of fake network packets is a potential approach to follow in defense mechanisms.

FAKE-A obfuscation approach presented similar performance results for both *PS* and *FS Scenarios*. Fig. 8 presents the FAKE-A results in *FS Scenario*. The 600-FAKE level also achieved the best results on masking the traffic from IoT devices. To evaluate the *FS Scenario*, we created dynamic port numbers and IP addresses to possibly create the flow 5-tuple (same source and destination IP addresses, and same source and port numbers). Once we generated fake traffic packets without changing any traffic feature value, the FAKE-A did not affect any specific network traffic feature considered in the scenarios. Then, it did not present different results for masking traffic features. For instance, in FAKE-A, we did not change the packet size values of the fake traffic. This justifies the similar results in both scenarios.



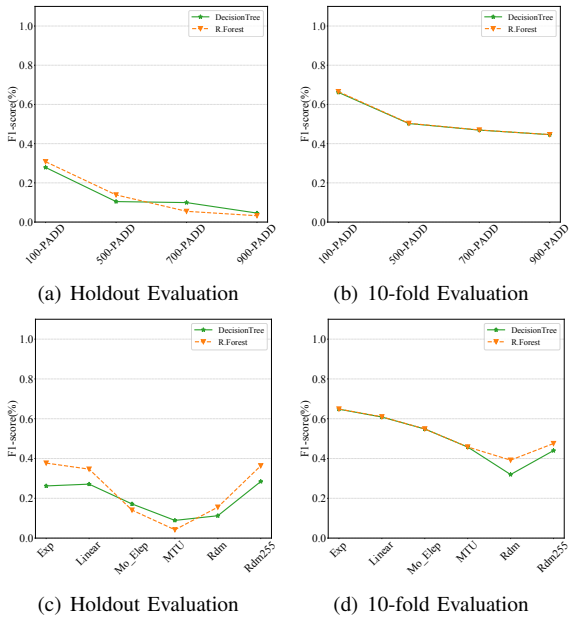
(a) 600-FAKE Level (b) 1000-FAKE Level

(c) 5000-FAKE Level (d) Rdm-FAKE Level

Fig. 8. FS: FAKE-A Approach Results

Once the PADD-A approach from [19] followed only the network packet-level scenario in their analysis, the following results present the obfuscation performance considering only the *PS scenario*. Moreover, we considered decision tree-based classifiers as [16], [19]. Note that CART and Decision Tree were implemented using the Decision Tree Classifier, which allows considering both classifiers as similar. The classifiers followed the holdout (testing and training dataset) and 10-fold cross-validation evaluations. We defined parameter $k = 10$ for the cross-validation according to [19].

Fig. 9 shows results under the packet padding by [19]. Figs. 9(a) and 9(b) present the PADD-A proposed by [19], while Figs. 9(c) and 9(d) display the well-established packet padding approaches (exponential, linear, mouse/elephants, MTU, random, and random 255). The holdout evaluation presented a better performance on reducing the success of the classifiers in identifying IoT devices. The 900-PADD and MTU achieved the best obfuscation results, reducing the F1-Score to $\approx 7\%$. This behavior was expected because the higher the packet padding level, the more uniform the packet sizes are, *i.e.*, probably all the network packets will have the same size. However, even the lowest level of packet padding, 100-PADD, reduced the performance of the classifiers to $\approx 32\%$, a significant performance degradation. One of the factors that may explain this performance degradation is that most IoT devices from the dataset generated packets with values around 100 bytes. Because of this, 500-PADD also considerably reduces the performance of the classifiers ($\approx 15\%$).



(a) Holdout Evaluation (b) 10-fold Evaluation

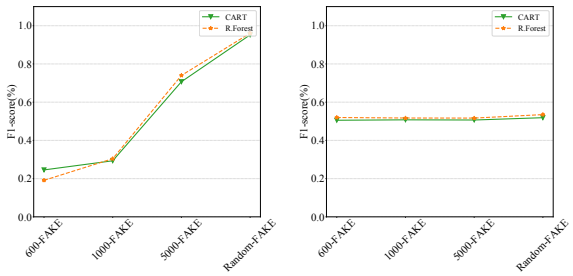
(c) Holdout Evaluation (d) 10-fold Evaluation

Fig. 9. PS: Packet Padding Approaches

As the goal of a network traffic obfuscation approach is to reduce the performance of classifiers, the 10-fold cross-validation presented the worst results compared to the holdout evaluation. The 10-fold evaluation tests more data combinations since it splits the dataset into 10 groups to classify the devices. In contrast, holdout only tests the training dataset with the testing dataset. Thus, even padding the packet size of IoT traffic to difficult the classifiers, the 10-fold still achieved considerable classification results. The 100-PADD level and Exponential presented $\approx 62\%$ of F1-Score. Even the 900-PADD level obtained 43% of F1-Score, similar to the 600-FAKE results. Therefore, the holdout evaluation with 900-PADD and MTU achieved the best obfuscation results. Nonetheless, they also increase the network bandwidth usage because the packet size has values around 1500 bytes. The highest value possible to a network packet. In contrast, the 100-PADD and 600-FAKE, and even the 600-FAKE and 500-PADD, added fewer bytes in the network bandwidth, which is more feasible for IoT environments.

To be a fair comparison between the fake traffic injection (FAKE-A) and packet padding (PADD-A) approaches, we generate classification results for the FAKE-A only con-

sidering the packet size and its statistics properties (mean, standard deviation, min, max, sum, and median). As [19] only used packet size as a network feature. We removed the other features from the analysis. Note that the packet size was pointed out as one of the most important features by the feature exploration. Hence, the classification results without defense approach, considering only the packet size (see Fig. 6(b)), achieved 69% and 70% of F1-Score in holdout evaluation for CART and R.Forest, respectively. In 10-fold cross-validation, the classifiers obtained ≈ 50% of F1-Score. Figs. 10(a) and 10(b) present the FAKE-A approach results for holdout and 10-fold evaluations. The results in Fig. 10(a) show that the FAKE-A using the 600-FAKE level can significantly reduce the performance of the analyzed classifiers. Compared to the results without defense approaches, the 600-FAKE in holdout reduced from 70% to ≈ 20% of F1-Score, a very similar result to the 100-PADD level from packet padding. The 600-FAKE also reduced the performance of the classifiers.



(a) Holdout Evaluation (b) 10-fold Evaluation

Fig. 10. FAKE-A Results Considering only Size-based Traffic Features

In contrast, the 1000-FAKE and Rdm-FAKE improved the results of the classifiers because they increased the number of network traffic samples. Table V shows a network overhead comparison, where PADD-A increased the network overhead when compared to FAKE-A. 1000-FAKE level increases only 0.8% of the network overhead compared to the original traffic without obfuscation. Based on the results presented, we can conclude that the fake traffic injection approach in the IoT environment did not require a massive amount of fake traffic since the IoT devices generate a tiny amount of traffic. Moreover, typical IoT devices generally send small packet sizes, ≈ 50 or 80 bytes. Thus, adding a small amount of fake network traffic is a good approach to obfuscate the traffic from such devices. Such a conclusion is a good direction for the IoT environment since it comprises devices with low processing and energy power. Adding much fake traffic (Rdm-FAKE) or increasing the size of the packet to a considerable size (MTU and 900-PADD) can harm network bandwidth. Nonetheless, although the significant performance degradation presented by FAKE-A and PADD-A in the lowest levels, the adversary still can identify devices. Thus, a combination of both approaches can be a direction to improve the obfuscation of IoT devices.

C. Discussion and Limitations

Based on the results, PCA and FA have highlighted the features that better contribute in the representation of the network traffic. Such output gives insight on what feature to use in traffic obfuscation mechanisms. However, network traffic obfuscation approaches either send additional traffic to hide real traffic or pad the packet size. Hence, such approaches consume additional bandwidth, computational power and energy, increasing computational cost. In IoT such additional consumption can be critical. However, the bandwidth and computing resources consumed by IoT traffic is relatively small due to the limited functionality of IoT devices. As shown in our obfuscation results, the amount of fake traffic needed to mask IoT traffic is very small. Even for IoT devices with high bandwidth consumption, such as monitoring cameras and smoke sensors, the low levels from the PADD-A and FAKE-A hide such traffic without increasing the amount of traffic in a huge proportion. Therefore, the fake traffic injected is reasonably limited and does not incur high additional costs. The PADD-A, depending on the length of the packets, can add high additional costs. In IoT, it is crucial to follow low levels of obfuscation to not increase the number of bytes. Analyzing the network volume to send additional traffic when idle also can be a solution. However, the in-home user does not have enough knowledge to control when to send additional fake traffic. Thus, an automatic and hybrid (with PADD-A and FAKE-A) approach to change the amount of traffic according to the network behavior can be a solution to avoid consuming resources inefficiently.

TABLE V
NETWORK OVERHEAD COMPARISON OF OBFUSCATION LEVELS

Traffic Type	Traffic Size	Netw. Overhead (%)
Original Traffic	269.41 MB	0%
600-FAKE	269.79 MB	**0.13**%
1000-FAKE	271.65 MB	**0.82**%
500-PADD [19]	1622.9 MB	602.2%
700-PADD [19]	2271.49 MB	843.1%

VI. CONCLUSION AND FUTURE WORKS

This article introduced the IoTReGuard, which reveals and guards IoT network traffic features. It innovates by exploring network traffic features to reveal the key ones and mask them to protect users' privacy. The method evaluation has followed a trace-driven approach, employing an IoT dataset. By IoT network feature exploration and data instrumentation, IoTReGuard provides valuable information on network traffic features to mask them. Results showed that IoTReGuard reduced from 70% to 20% of F1-Score on identifying IoT devices, improving user privacy. In future works, we intend to analyze the proposed method in an online IoT environment.

ACKNOWLEDGMENT

The authors thank the agencies FAPESP/Brazil, under grants #2018/23098-0 and 2021/06733-6, CNPq/Brazil, under grants #32204/2018-0 and 313844/2020-8, and CAPES/Brazil.

REFERENCES

[1] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, and A. Erbad, "Edge computing for smart health: Context-aware approaches, opportunities, and challenges," *IEEE Netw.*, vol. 33, no. 3, pp. 196–203, 2019.

[2] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1646–1685, 2020.

[3] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1191–1221, 2020.

[4] Statista, "Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)," Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, 2016, accessed February, 2022.

[5] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" in *Proc. of Security Privacy in Wireless Mobile Netw.* New York, USA: ACM, 2020, p. 207–218.

[6] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 2019.

[7] D. Saha, "How the world became data-driven, and what's next," Available: https://www.forbes.com/sites/googlecloud/2020/05/20/how-the-world-became-data-driven-and-whats-next/?sh=4b7508b657fc, 2020, accessed February, 2022.

[8] I. H. Sarker, "Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective," *Springer Nature Computer Sci.*, vol. 2, no. 5, p. 377, 2021.

[9] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 63–78, 2018.

[10] Y. Zhang and A. Årvidsson, "Understanding the characteristics of cellular data traffic," in *Proc. of the SIGCOMM Workshop on Cellular Netw.* ACM, 2012, pp. 13–18.

[11] L. Vu, H. V. Thuy, Q. U. Nguyen, T. N. Ngoc, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Time series analysis for encrypted traffic classification: A deep learning approach," in *Proc. of the Symposium on Commun. Inf. Technol.* IEEE, 2018, pp. 121–126.

[12] P. Maxwell, E. Alhajjar, and N. D. Bastian, "Intelligent feature engineering for cybersecurity," in *Proc. of the International Conference on Big Data*. IEEE, 2019, pp. 5005–5011.

[13] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "ProfilIoT: A machine learning approach for iot device identification based on network traffic analysis," in *Proc. of the Symposium on Applied Comput.* ACM, 2017, p. 506–509.

[14] P. Maxwell, D. Niblick, and D. C. Ruiz, "Using side channel information and artificial intelligence for malware detection," in *Proc. of the International Conference on Artificial Intell. Comput. Appl.* IEEE, 2021, pp. 408–413.

[15] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 45–59, 2020.

[16] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," in *Proc. of the Symposium on Security Privacy*. IEEE, 2012, pp. 332–346.

[17] S. Feghhi and D. J. Leith, "An efficient web traffic defence against timing-analysis attacks," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 525–540, 2018.

[18] I. Hafeez, M. Antikainen, and S. Tarkoma, "Protecting IoT-environments against traffic analysis attacks with traffic morphing," in *Proc. of the PerCom Workshops*. IEEE, 2019, pp. 196–201.

[19] A. J. Pinheiro, P. F. de Araujo-Filho, J. d. M. Bezerra, and D. R. Campelo, "Adaptive packet padding approach for smart home networks: A tradeoff between privacy and performance," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3930–3938, 2020.

[20] L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "App traffic mutation: toward defending against mobile statistical traffic analysis," in *Proc. of the INFOCOM*. IEEE, 2018, pp. 27–32.

[21] ——, "Optimal packet camouflage against traffic analysis," *ACM Trans. Privacy Security*, vol. 24, no. 3, pp. 1–23, 2021.

[22] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, 2018.

[23] A. Vergütz, I. Medeiros, D. Rosário, E. Cerqueira, A. Santos, and M. Nogueira, "A method for identifying ehealth applications using side-channel information," in *Proc. of GLOBECOM*. IEEE, 2019, pp. 1–6.

[24] A. Vergutz, G. Noubir, and M. Nogueira, "Reliability for smart healthcare: A network slicing perspective," *IEEE Netw.*, vol. 34, no. 4, pp. 91–97, 2020.

[25] N. Prates, A. Vergütz, R. T. Macedo, A. Santos, and M. Nogueira, "A defense mechanism for timing-based side-channel attacks on IoT traffic," in *Proc. of GLOBECOM*. IEEE, 2020, pp. 1–6.

[26] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The dark side (-channel) of mobile devices: A survey on network traffic analysis," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2658–2713, 2018.

[27] I. Ullah, K. Mengersen, R. J. Hyndman, and J. McGree, "Detection of cybersecurity attacks through analysis of web browsing activities using principal component analysis," *arXiv preprint arXiv:2107.12592*, 2021.

[28] D. H. Hoang and H. D. Nguyen, "A PCA-based method for IoT network traffic anomaly detection," in *Proc. of the International Conference on Advanced Commun. Technol.* IEEE, 2018, pp. 381–386.

[29] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," *Netw. and Distrib. Syst. Security Symposium*, vol. 2020.

[30] A. Pekar, J. Mocnej, W. K. G. Seah, and I. Zolotova, "Application domain-based overview of IoT network traffic characteristics," *ACM Comput. Surveys*, vol. 53, no. 4, Jul. 2020.

[31] D. Y. Huang, N. Apthorpe, F. Li, G. Acar, and N. Feamster, "IoT inspector: Crowdsourcing labeled network traffic from smart home devices at scale," *ACM Interactive, Mobile, Wearable and Ubiquitous Technol.*, vol. 4, no. 2, Jun. 2020.

[32] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Traffic classification of mobile apps through multi-classification," in *Proc. of GLOBECOM*. IEEE, 2017, pp. 1–6.

[33] P. Azad, N. J. Navimipour, A. M. Rahmani, and A. Sharifi, "The role of structured and unstructured data managing mechanisms in the internet of things," *Cluster Comput.*, vol. 23, no. 2, pp. 1185–1198, 2019.

[34] S. Verma, K. Jain, and C. Prakash, "An unstructured to structured data conversion using machine learning algorithm in internet of things (IoT)," in *Proc. of the Int. Conf. on Innovative Comput. Commun.*, 2020.

[35] R. Hou, Y. Kong, B. Cai, and H. Liu, "Unstructured big data analysis algorithm and simulation of internet of things based on machine learning," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 5399–5407, 2020.

[36] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 2008.

[37] A. C. Callado, C. A. Kamienski, G. Szabó, B. P. Gero, J. Kelner, S. F. Fernandes, and D. F. H. Sadok, "A survey on internet traffic identification." *IEEE Commun. Surveys Tuts.*, vol. 11, no. 3, pp. 37–52, 2009.

[38] J. S. Atkinson, J. E. Mitchell, M. Rio, and G. Matich, "Your WiFi is leaking: What do your mobile apps gossip about you?" *Elsevier Future Generation Comp. Syst.*, vol. 80, pp. 546–557, 2018.

[39] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[40] J. He, Q. Xiao, P. He, and M. S. Pathan, "An adaptive privacy protection method for smart home environments using supervised learning," *Future Internet*, vol. 9, no. 1, p. 7, 2017.

[41] N. Apthorpe, D. Yuxing Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart (er) IoT traffic shaping," *arXiv e-prints*, pp. arXiv–1812, 2018.

[42] K. Yu, Q. Li, D. Chen, M. Rahman, and S. Wang, "PrivacyGuard: Enhancing smart home user privacy," in *Proc. of the International Conference on Inf. Process. Sensor Netw.* ACM, 2021, p. 62–76.

[43] A. Diyanat, A. Khonsari, and S. P. Shariatpanahi, "A dummy-based approach for preserving source rate privacy," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1321–1332, 2016.

[44] X. Fu, B. Graham, R. Bettati, and W. Zhao, "On effectiveness of link padding for statistical traffic analysis attacks," in *Proc. of Conference on Distrib. Comp. Syst.* IEEE, 2003, pp. 340–347.

[45] M. Shen, Y. Liu, L. Zhu, K. Xu, X. Du, and N. Guizani, "Optimizing feature selection for efficient encrypted traffic classification: A systematic approach," *IEEE Netw.*, vol. 34, no. 4, pp. 20–27, 2020.

[46] R. Abdulhammed, M. Faezipour, H. Musafer, and A. Abuzneid, "Efficient network intrusion detection using PCA-based dimensionality reduction of features," in *Proc. of Symposium on Netw. Comp. Commun.* IEEE, 2019, pp. 1–6.

[47] Y. Yao, J. R. Basdeo, S. Kaushik, and Y. Wang, "Defending my castle: A co-design study of privacy mechanisms for smart homes," in *Proc. of Conference Human Factors Comput. Syst.* ACM, 2019, p. 1–12.

[48] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–45, 2017.
[49] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1686–1721, 2020.

Andressa Vergütz holds M.Sc. and Ph.D. degrees in Computer Science from Federal University of Paraná, Brazil. She was a visiting research student at University of Ottawa, Canada, in 2020. Her research interests include smart healthcare, network traffic analysis, network performance management, IoT, and network security. She is a member of Wireless and Advanced Networks research team and Center for Computational Security sCience.

Bruna V. Santos is Information Systems student at the Federal University of Santa Maria - Campus Frederico Westphalen (UFSM-FW). Her research interests include smart home, network traffic analysis, network security and IoT. She is a student member of the Brazilian Computer Society (SBC) and IEEE Communications Society (ComSoc).

Burak Kantarci (Senior Member, IEEE) received the Ph.D degree in computer engineering in 2009. He is an Associate Professor and the Founding Director of Smart Connected Vehicles Innovation Centre (SCVIC), and the Founding Director of the Next Generation Communications and Computing Networks (NEXTCON) Research Lab, University of Ottawa. He has coauthored over 250 publications in established journals and conferences, and 15 book chapters. In 2022, he was awarded a Minister's Award of Excellence in Innovation and Entrepreneurship from Ontario Ministry of Colleges and Universities. He served as the Chair of IEEE Communications Systems Integration and Modeling Technical Committee, and has served as the Technical Program Co-Chair/Symposium Co-chair of more than twenty international conferences/symposia/workshops, including IEEE Global Communications Conference (GLOBECOM)—Communications Systems QoS, Reliability and Modeling (CQRM) symposium. In 2021, he has been elected as the new Secretary of IEEE Social Networks Technical Committee. He is an Editor of the IEEE Communications Surveys & Tutorials, IEEE Internet of Things Journal, Vehicular Communications (Elsevier), and an Associate Editor for IEEE Networking Letters, and Journal of Cybersecurity and Privacy. He is Editor for several IEEE and Elsevier journals. He was a Distinguished Speaker of the ACM in 2019-2021. He is currently a Distinguished Lecturer of the IEEE Communications Society and IEEE Systems Council.

Michele Nogueira is an associate professor in the computer science department of Federal University of Minas Gerais. She holds a Doctorate degree in Computer Science from the UPMC-Sorbonne University, Laboratoire d'Informatique de Paris VI (LIP6). She was in a Sabbatical Leave at Carnegie Mellon University in 2016-2017. Her research interests include wireless networks, network security and resilience. She was an Associate Technical Editor for the IEEE Communications Magazine and the Journal of Network and Systems Management.