What-App? App Usage Detection Using Encrypted LTE/5G Traffic

Jinjin Wang*
University of Birmingham
Birmingham, UK
jxw1663@student.bham.ac.uk

Zishuai Cheng*
Beijing University of Posts and Telecommunications
Bejing, China
chengzishuai@bupt.edu.cn

Mihai Ordean†
University of Birmingham
Birmingham, UK
m.ordean@bham.ac.uk

Baojiang Cui
Beijing University of Posts and Telecommunications
Bejing, China
cuibj@bupt.edu.cn

Abstract

Cellular traffic fingerprinting attacks, in which an unprivileged adversary passively monitors encrypted wireless channels to infer user activities, introduce significant privacy risks by giving attackers the ability to track user behaviors, infer sensitive activities, and profile victims without authorization. Although such attacks have been discussed for LTE and 5G, many existing studies rely on idealized assumptions that fall short when faced with the complexities of real-world practical scenarios.

In this paper, we present the first practical traffic fingerprinting attack leveraging a Man-in-the-Middle (MITM) Relay in an operational cellular network. Implemented with open-source software, our attack allows a passive adversary to identify user applications with up to 99.02% accuracy, even under noisy conditions. We evaluate our method using 40 applications across five categories on multiple COTS user equipment (UE). Our approach further demonstrates the ability to infer fine-grained user activities such as browsing, messaging, and video streaming under practical constraints, including partial traffic knowledge and app version drift. The attack also achieves cross-device and cross-network transferability, and it remains robust in open-world scenarios where only a subset of application traffic is known to the adversary.

We additionally propose a novel traffic regularization-based defense tailored specifically for cellular networks. This defense operates as an optional, backward-compatible security layer integrated seamlessly into the existing cellular protocol stack, effectively balancing security strength with practical considerations such as latency and bandwidth overhead.

Keywords

Cellular Network Security, 5G and LTE security and privacy, Traffic Fingerprinting, Man-in-the-Middle (MITM) Relay, False Base Station, Fingerprinting Defense, Traffic Regularization

*Both authors contributed equally to this research.
†Corresponding author

Proceedings on Privacy Enhancing Technologies 2026(1), 242–256

https://doi.org/10.56553/popets-2026-0013

1 Introduction

With the rapid advancement of cellular technologies such as LTE and 5G, users increasingly prefer cellular networks over Wi-Fi hotspots, particularly in public venues like airports and cafes when conducting sensitive activities (e.g., mobile banking and online payments). This preference largely stems from the perception that cellular infrastructure, operated and secured by network providers, offers enhanced protection [11]. However, this perceived security is misleading. Despite robust infrastructural protection and strong encryption protocols, the openness of wireless channels exposes cellular networks to privacy breaches.

Recent studies have demonstrated that adversaries can exploit side-channel information and machine learning techniques to derive sensitive user data (such as application usage and browsing behavior) from encrypted traffic. Multiple studies have shown that user activity can be inferred by leveraging the physical scheduling information available over the air [6, 23, 27, 42]. Furthermore, Man-in-the-Middle (MITM) Relay attacks, as discussed in [4, 19, 33], enable adversaries to intercept the entirety of encrypted communications without access to encryption keys. This intercepted traffic can be used to extract fingerprints and execute selective denial-of-service (DoS) attacks [10, 19, 22, 33, 47].

Unfortunately, current app fingerprinting studies in cellular networks often fail in realistic environments due to idealized assumptions. First, these studies often exclude realistic attack conditions, relying on controlled experimental data. Second, their analysis of attack scenarios tends to be simplistic; for example, traffic is typically collected during the app launch phase, without considering concurrent background activity. In real-world mobile traffic, data is interwoven between multiple foreground and background applications, encrypted across several protocol layers, and lacks explicit markers for session initiation. Third, the effects of app versions and device models on fingerprinting accuracy are often underexplored. While some studies have examined these factors [6], none offer a comprehensive evaluation that reflects actual attack conditions.

In response to these limitations, we design and implement a practical, real-world attack scenario on both 4G and 5G standalone (5G-SA) networks. By setting up a MITM Relay to intercept and analyze encrypted traffic between user equipment (UE) and the core network, we gather encrypted traffic traces and train a Convolutional Neural Network (CNN) based classifier for accurate app fingerprinting. We collect data using three commercial off-the-shelf (COTS) UEs running the 40 most popular mobile applications across five

categories [1]. Moving beyond simulation-based studies, we assess the impact of background traffic, app version, and device model variations on attack performance and further validate the attack's feasibility and effectiveness in commercial networks. Our results demonstrate that these attacks can not only accurately identify which applications users are operating, but also distinguish between specific in-app functionalities. This finding highlights the real privacy risks associated with these attacks.

To address the growing threat of mobile traffic fingerprinting, we analyze the traffic characteristics that contribute most to upper-layer information leakage. Our experiments further reveal that simplistic defenses, such as solely obfuscating packet sizes or delaying packets, are insufficient to mitigate these attacks. These results are consistent with prior research in web fingerprinting [9, 12, 46], indicating that inadequately designed defenses may only provide a false sense of security.

Motivated by these observations, we propose a novel defense mechanism that is integrated into the Packet Data Convergence Protocol (PDCP) layer. Inspired by TAMARAW [9], our defense employs a regularization-based shaping mechanism that offers strong security guarantees. By classifying applications based on their latency requirements and traffic throughput, our approach minimizes defense overhead while ensuring robust security.

Contributions. This paper makes the following contributions.

- We present the first comprehensive evaluation of fingerprinting attacks in real-world LTE and 5G-SA networks using a MITM Relay.
- We propose and implement a novel defense scheme for cellular networks that minimizes overhead for a given security level, tailored to the latency tolerance of applications.
- We validate our methods on multiple commercial cellular networks, testing 49 app activities across 40 popular Android applications, and achieving a classification accuracy exceeding 99.02%.

Organization. Section 2 provides background on fingerprinting techniques and related defense mechanisms. Section 3 presents the adversary model. In Section 4, we detail the attack design and evaluation. Section 5 introduces our proposed defense mechanism and its evaluation. Finally, Section 6 concludes the paper.

2 Background and Related Work

2.1 LTE and 5G Network Architectures

LTE and 5G networks consist of three main components: the user equipment (UE), the cellular base station (BS) (known as eNodeB in LTE or gNodeB in 5G), and the core network. Communication between the BS and the core is secured via IPsec over a wired infrastructure; in contrast, the over-the-air radio link between the UE and the BS remains susceptible to eavesdropping and tampering. To safeguard user traffic, the Packet Data Convergence Protocol (PDCP) layer encrypts IP packets using symmetric cryptography (typically an AES-CTR-based cipher), thereby protecting the payload. However, not all transmission-related information is concealed; details such as the Radio Network Temporary Identifier (RNTI), payload size, packet direction, and arrival time remain accessible through the radio link layer, providing valuable side-channel data that can be exploited for traffic analysis and fingerprinting attacks.

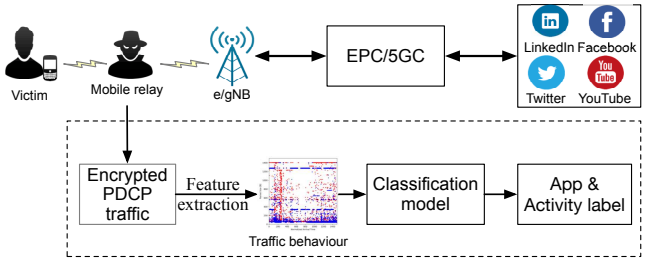


Figure 1: Attack framework. The attacker sets up a position between the victim and the e/gNB to capture traffic (up). This traffic is processed in order to identify the apps and their activities as used on the victim UE (down).

2.2 Identity Mapping in Cellular Networks

A radio link-layer adversary can distinguish a user's signals based on the RNTI associated with each transmitted signal block. The RNTI is a temporary identifier assigned by the BS to a UE for the duration of a radio connection, and is used to uniquely identify the UE's signaling and data transmissions within a cell. For long-term victim monitoring, the adversary must map the victim's Personally Identifiable Information (PII), such as an International Mobile Subscriber Identity (IMSI), social media account, or phone number, to the RNTIs currently used by the victim. In 4G networks, a victim's IMSI can be mapped to the corresponding RNTI by means of spoofing and intercepting unprotected unicast messages, such as *Identity Request* and *Identity Response*. In both 4G and 5G networks, even without access to the IMSI, the adversary can still infer the victim's RNTI if they know another identifier (e.g., a social media account or phone number) associated with the target. In this case, the adversary sends stealthy, patterned traffic toward the target device, and then correlates the resulting network activity across all observed users to identify the RNTI corresponding to the victim [10, 27].

2.3 MITM Relay Attacks in Cellular Networks

MITM Relay attacks exploit structural weaknesses in the LTE/5G access architecture, particularly the lack of security enforcement at the physical and data link layers of the air interface between the UEs and the BSs. Specifically, these attacks leverage the absence of physical channel binding during the initial network attachment procedure, allowing an adversary to relay control signaling and user-plane data between targeted UEs and the network without possessing any legitimate credentials.

In this class of attacks, the adversary takes a man-in-the-middle position as a relay node between the UE and the BS (see fig. 1). This relay transparently forwards all messages exchanged during the Authentication and Key Agreement (AKA) procedure. As the AKA protocol validates cryptographic credentials at higher protocol layers, such as PDCP and Non-Access Stratum (NAS) layer, but does not perform any verification at the physical layer, the core network completes mutual authentication and establishes a security context under the assumption that it is communicating directly with the legitimate UE. Since in LTE and 5G, security protections apply only at the PDCP layer and above, lower layers such as the Radio Link

Table 1: Comparison with existing cellular network fingerprinting works.

Paper	Bkg. Noise	App Ver.	Device Model	Commercial Net	Open-world	Target
Zhai et al. [47]	✗	✗	✓	✗	✗	20 apps
Trinh et al. [42]	✓	✗	✗	✓	✓	6 apps
Kohls et al. [22]	✓	✗	✓	✓	✗	Websites
Bae et al. [6]	✗	✗	✓	✓	✓	Video
Lakshmanan et al. [24]	✗	✗	✓	✓	✓	Video
Baek et al. [7]	✓	✗	✓	✓	✗	9 apps
Islam et al. [20]	✗	✗	✗	✗	✗	12 apps
Our Work	✓	✓	✓	✓	✓	40 apps / 49 activities

Note: **Bkg. Noise** indicates whether background traffic from idle/system apps is considered. **App Ver.** means the impact of different application versions is analyzed. **Device Model** reflects whether different smartphones were evaluated. **Commercial Net** Indicates if evaluation is conducted using data captured by an adversary on commercial networks. **Target** lists the attack classification goal (e.g., apps, websites). **Open-world** means the work evaluated unseen app traffic. ✓: factor considered; ✗: not considered.

Control (RLC), Medium Access Control (MAC), and Physical (PHY) layers lack integrity protection and source authentication, thereby allowing the adversary to persistently maintain the connection through these insecure layers.

A characteristic of these attacks is that the adversary does not need to compromise cryptographic material, break encryption schemes, or manipulate authentication messages. Instead, these attacks enable eavesdropping on the traffic simply by relaying messages in a protocol-compliant manner. As a result, the attacker gains real-time access to encrypted communication traffic between the UEs and the network, which provides a foundation for subsequent attacks such as user tracking, traffic injection, data manipulation, or service redirection [19, 33, 34].

2.4 Traffic Fingerprinting Attacks

2.4.1 Network and Transport Layer Fingerprinting. Traffic fingerprinting attacks at the network and transport layers aim to infer sensitive user activities from encrypted traffic by exploiting information contained in IP packets and upper-layer packet headers [15, 26, 31, 35, 36, 38]. Typically, attackers collect and label traffic sequences corresponding to target webpages or videos, then employ machine learning or deep learning techniques with pretrained models to analyze intercepted victim traffic and identify detailed online activities, such as the specific webpages visited or video titles accessed. Although these attacks can effectively reveal fine-grained user behaviors, they often rely on a strong assumption: privileged access to network or transport layer metadata (e.g., IP addresses and TCP/UDP port numbers). This assumption, however, is generally not met in cellular networks, where app-to-server communications are secured both at the IP level with TLS and at the PDCP layer with an AES-based stream cipher (EEA2).

2.4.2 Traffic Fingerprinting on Cellular Networks. Similar to fingerprinting attacks at the network and transport layers, attackers in cellular networks exploit traffic characteristics to infer sensitive user activities. However, due to the inherent security measures in cellular infrastructures, adversaries typically cannot access network-layer data directly with sufficient depth. Nonetheless, recent research has demonstrated that attackers can still infer user activities by leveraging information leaked from insecure radio links. For instance, Bae et al. [6] successfully identified the specific videos a victim was watching by analyzing information disclosed through the PDCP data stream; however, these attacks were confined to video content and did not extend to other types of user activities. Similarly, Baek et al. [7] employed sniffing tools (such as OWL and LTESniffer [8, 13, 16]) to capture physical-layer scheduling information (i.e., Downlink Control Information, DCI) for application identification, although these experiments did not systematically evaluate the influence of real-world conditions or open-world scenarios on identification performance. In table 1, we summarize and compare existing state-of-the-art work in cellular network fingerprinting with our work. We would like to highlight that, while the aforementioned studies collectively demonstrate the feasibility of fingerprinting attacks in cellular networks, they generally fall short of providing a systematic evaluation under realistic conditions.

2.5 Traffic Fingerprinting Defenses

Current traffic fingerprinting defenses have primarily targeted application-layer protocols such as HTTP and website traffic. A variety of techniques have been proposed to mitigate such attacks. For instance, traffic randomization-based defenses introduce dummy packets and transmission delays to increase intra-class variance and thereby reduce classifier confidence [5, 14, 21, 28, 30]. In addition, adversarial machine learning-based defenses craft carefully designed perturbations using techniques such as GANs or gradient-based methods so as to deliberately induce misclassifications by the attackers' models [5, 18, 29, 32]. Despite the promise of these approaches, they have several critical shortcomings. First, both randomization-based and adversarial defenses are typically evaluated only against known attack models, and as a result, they may fail to generalize to continuously evolving attackers who can rapidly update their classifiers to bypass these perturbations [46]. This limitation renders these defenses less suitable for cellular network infrastructures. Second, application traffic in mobile networks exhibits significant heterogeneity across dimensions such as packet size distributions, inter-packet timings, burstiness, throughput, and downlink and uplink ratios. Such variation can substantially undermine the effectiveness of these defenses [12], particularly for approaches lacking formal guarantees. In contrast, traffic regularization-based defenses reshape the original application traffic into standardized patterns, such as fixed transition rates and packet sizes, thereby providing reliable security protection by rendering network traces

from different sources indistinguishable [9, 12, 17, 43, 44]. However, this approach introduces additional bandwidth and latency overheads. As we will show in section 5, our defense model builds upon these regularization-based techniques while explicitly addressing the trade-off between latency and bandwidth overheads under a fixed security level.

3 Threat Model

We consider a passive link-layer adversary who monitors the encrypted wireless communication between the user equipment (UE) and the cellular base station (e.g., eNB or gNB). The adversary is assumed to possess at least one piece of the victim's PII, and could selectively intercept the victim's PDCP traffic over the air interface using the MITM Relay (see section 2.2). However, the adversary cannot decrypt, modify, or inject any packets. He has no access to the core network or IP-layer data. The adversary can observe traffic metadata such as packet sizes, directions, and timing information. Additionally, the adversary has full knowledge of the traffic patterns of a set of target applications, obtained through offline profiling. The adversary is also able to train classifiers offline using labeled traffic traces and use them during the attack phase.

With the knowledge from above, the adversary aims to identify the smartphone apps installed on a victim's device and their use patterns from the predefined list of targeted apps. A basic example of this attack could be an adversary that learns that its victim sent or received several WhatsApp messages, or was watching YouTube while also texting on WhatsApp. In more critical scenarios, the attacker can precisely identify one or more applications and use this insight as an entry point for further exploitation. For instance, if the adversary detects that the victim is actively using both a delivery app and banking software at the same time, they can infer that the victim is currently awaiting a delivery. This information could then lead to exposure of the victim's identity or make them vulnerable to targeted threats.

4 Attack Design and Evaluation

In this section we give a detailed description of our attack, followed by our experimental setup and evaluation.

4.1 Attack Design

As stated, the goal of our adversary is: (1) to identify a set of smartphone applications installed on a victim's device, and (2) to infer user activities within these applications with a high level of detail. To achieve these objectives, the attack proceeds in two stages. In the *attack preparation stage*, the adversary collects application-specific traffic in a controlled environment, extracts relevant features, and trains a classification model capable of recognizing different apps and activity types. In the *activity identification stage*, the adversary passively captures encrypted traffic from the victim over the wireless channel using a MITM Relay; then infers the victim's apps and specific in-app behaviors using the pre-trained model.

4.1.1 Attack Preparation Stage. In the preparation stage, the adversary collects app traffic traces and uses them to train a classification model. It is important that this data shares the same distributional characteristics as the traffic observed during the attack phase. As such, given that our adversary is interacting with the network at the PDCP layer, and is only able to observe encrypted traffic, the relevant data for the model will be restricted to PDCP packet metadata consisting primarily of packet-sizes and packet-arrival-times. There are multiple practical approaches for obtaining such training data, as described in the following.

An initial direct method for collecting training data for fingerprinting relies on an attacker controlling a UE connected to a private 5G/LTE testbed. This setup can be deployed using open-source software (e.g., Open5GS, srsRAN). With this method, PDCP metadata can be directly captured during app execution, with no need for further modification.

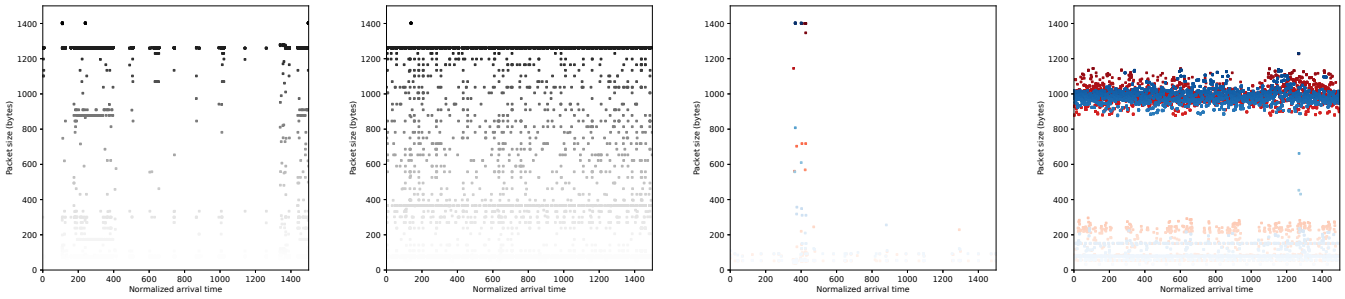
Alternatively, in a slightly modified method that better emulates the real-world attack setting, the adversary collects traffic through a MITM Relay, which intercepts the communication between the UE and the gNB. This approach more closely reflects the passive interception scenario and avoids the potential time distortions introduced by the relay during the attack stage.

Finally, due to the use of symmetric encryption between the network and PDCP layers, traffic characteristics at the PDCP layer are correlated with those at the IP layer. Leveraging this property, the adversary can adopt a lightweight approach by extracting training data from IP-layer traffic and mapping it to the PDCP layer. While convenient, this method requires careful handling: in addition to target application traffic, background traffic must also be captured or generated to reflect realistic environmental noise. Naturally, training data collected in this way may differ slightly from traffic observed at the PDCP layer during attacks.

Fingerprint Extraction. Traffic patterns generated by different app activities often exhibit distinguishable temporal and size distributions (see fig. 2 and fig. A1). However, real-world traffic introduces several challenges: user behaviors are asynchronous and session-dependent; background traffic (e.g., keep-alive packets, notifications) is mixed with target traffic; and content-specific differences (e.g., different YouTube videos) add further unpredictability.

To address these challenges, we adopt a fingerprint extraction method that preserves the complete data distribution and converts it into a two-dimensional representation. Then we apply a CNN model, inspired by [25, 40], as our primary classifier. The CNN is well-suited for this task, as it can effectively capture spatial and temporal correlations between packets in the transformed traffic data, enabling robust classification despite noise and variability.

Specifically, we extract fingerprints from collected traffic using a T-second long sliding window with an n-second step, where $n \ll T$. Data in each window is converted into a payload-size distribution (PSD) for a specific time interval. Then, we define a system of coordinates where the X-axis represents the observed time, and the Y-axis indicates the packet size similar to [31]. We additionally use colors to represent the direction of the traffic, i.e., red for uplink, blue for downlink, and black for links which do not contain directional information. We then generate the PSD figures by plotting all observed PDCP packets captured in the defined interval with the following set of adjustments. For the X-axis, we first normalize all the records in an interval by subtracting the initial timestamps. Then, we scale the X-axis to fit into the interval $[0, 1500]$ such that we obtain square figure representations of the PSD as shown in fig. 2.



(a) Instagram (b) Facebook (c) WhatsApp - text (d) WhatsApp - video

Figure 2: Example of PSD images, which provide a visual representation of traffic traces from four different activities across three applications (Instagram, Facebook, WhatsApp chat, and WhatsApp video call). In each figure, every packet is depicted as a single point, where the x-axis represents its arrival time and the y-axis indicates its size. The black-and-white graph represents traffic in only one direction, with the shade indicating the traffic density. Uplink and Downlink information are further distinguished by red and blue, respectively.

4.1.2 App Identification Stage. After completing the preparation stage, the adversary initiates the fingerprinting attack by deploying a MITM Relay within the victim's vicinity [1], as shown in fig. 1. Once the traffic relay is established, the adversary continuously monitors encrypted data exchanged between the victim UE and the network. The pretrained classifier is used with a sliding window of 60 seconds to infer user activity in real time. With this setup, the attacker's goal is to: (1) identify which applications from a predefined target list are in use; (2) determine how those applications are being used, for instance, distinguishing between browsing reels on Instagram and sending messages. With the traffic collected, the attacker can also estimate the (3) duration of each activity and (4) infer the temporal sequence of usage patterns.

Although a 60-second window provides a good trade-off between accuracy and latency, overlapping activities within a window can still occur. For example, a victim could be watching a YouTube video while simultaneously receiving WhatsApp messages. In such a case, if the adversary requires more precise traffic localization in time, or better separation between concurrent or interleaved app activities, then the accuracy can be adjusted through the size of the sliding window.

4.2 Attack Evaluation

In this section, we comprehensively evaluate the effectiveness and generalization of our fingerprinting attack across multiple dimensions. First, we compare the performance differences among multiple classifiers, including our CNN-based model and traditional models such as SVM, KNN, and MLP, and assess their robustness when only limited features are available, such as unidirectional traffic, isolated packet-size information, or timing information (see section 4.2.3). Next, we determine the classifier's accuracy in an open-world scenario where the attacker can access traffic from only a subset of applications (section 4.2.5). Then, we investigate the impact of application software versions, network types, and device models on classification accuracy in both commercial and private network environments [2] (sections 4.2.6 and 4.2.8). Finally, we analyze the effects of background traffic noise, as well as variations in signal strength and overall network connection quality, on attack performance (sections 4.2.7 and 4.2.11).

Ethical Consideration. We comply with the law and other users' privacy by controlling the transmission power of our cellular relay in order to avoid interference with commercial network nodes. When validating our attacks on commercial networks, we isolate both the rogue base station and the target User Equipment (UE) within a Faraday cage. Additionally, we configure our relay to allow connections only from our specific test SIM cards. This ensures that even if an external device detects our base station, it cannot establish a connection with our relay.

4.2.1 Experimental Setup. Our data acquisition setup consists of two USRP B210 SDRs, an Intel i9-12900 PC with 32G RAM running a modified version of srsRAN which supports the relay functionality. Traffic collection is done in the form of pcap files as seen by the cellular-relay node and consists of user-plane PDCP traffic and radio identifier (i.e., RNTI). The training stage is performed on a private LTE network using Open5GS and sysmoISIM-SJA2 programmable USIM cards for the test UEs.

We use three UEs: a Google Pixel 5, a Samsung S8, and an iPhone 12. The Pixel 5 runs Android 13 on a Qualcomm Snapdragon 765G chipset, and the Samsung S8 runs Android 8 on an Exynos 8895 chipset. When collecting the *training* and *test* datasets, the Pixel 5 is used with our private network and is controlled by ADB to facilitate the simulation task. For the *attack* datasets, both the Pixel 5 and the Samsung S8 are used to test with commercial LTE networks. The iPhone is used to connect to the commercial LTE network and a private 5G-SA network.

4.2.2 Data Collection. We collect the data using our MITM Relay from a private LTE network, as described in section 4.1.1. The data comes from the most popular Android applications from Android Rank [1]. We focus on five common application categories: video,

[1] Typically within 100 m, but can be up to 2 km with an amplifier [39].

[2] The classifier is trained on the private network.

music, social networking, communication, and gaming, which capture a broad range of typical user behaviors. Within each category, we select the top eight apps according to the Android Rank listings. Each app is mapped to one or more representative user activities, resulting in a total of 49 (app, activity) labels, as shown in table A2 in the Appendix. To generate realistic training traffic, we simulate these activities through automated scripts or manual interactions, including actions such as playing or pausing media, scrolling, sending messages, and playing games. For streaming and messaging apps, actions are randomized to increase variance; game traffic is collected via manual gameplay to preserve interaction richness. We also ensure that traffic includes foreground usage and background behaviors (e.g., notifications, keep-alive messages) to closely mimic real-world app usage patterns. Training data is collected with Pixel 5 on our private network by simulating the major activities of each app for 45 minutes. We collect about 29.92 GB of encrypted PDCP traffic, totaling 2205 minutes of use time.

4.2.3 Classifier Setup and Evaluation. To train our model, we extract traffic fingerprints from collected data as inputs to the CNN model using the approach described in section 4.1.1.

In order to determine the appropriate step size for training, we first conducted experiments on a small subset of applications to compare the impact of different step sizes on training performance (see fig. 5). Through experiments on five apps from different categories, we found that the impact of step size is most significant for Instagram Real, where a longer step size resulted in a decrease in classification accuracy of up to approximately 50%. Moreover, the accuracy across all apps is most balanced when the step size is set to 5. Therefore, we use a 60-second window size with a 5-second step size to extract 540 PSD samples, which include directional information. We then select 500 of these to evaluate the performance. As traffic is independently and uniformly generated, we perform a three-way split where we use 80% to train our model, and the remaining 10% and 10% to validate and test classification performance, respectively. We use the micro average F1 score to evaluate our model.

We also implement three traditional classifiers: Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP), following parameters from previous cellular network fingerprinting literature [47]. These models operate on handcrafted statistical features extracted from each fingerprint window (e.g., number of packets, percentile of packet sizes, and packet arrival times) [22, 47]. For the MLP, we further tune the hidden layer size to improve accuracy when using the full feature set. We provide hyperparameters and configurations for each model in the table 2. Our classification results are summarized in the table 3. The CNN achieved the highest accuracy (i.e., 99.02%) when using the full set of features, demonstrating its ability to capture complex traffic patterns. All other traditional models also achieved more than 95% classification accuracy.

4.2.4 Classifier Performance on Limited Features. In practical attack scenarios, adversaries may not have access to complete bidirectional traffic between the UE and the network. Some attackers can only obtain the downlink scheduling data [13, 27], which does not fully represent packet size details. Additionally, native defense mechanisms such as packet padding or intentional delays might be used to mitigate fingerprinting attacks [21, 28, 45]. Thus, we evaluate our classifier efficiency under scenarios with restricted traffic information: (1) uplink-only (UL) or downlink-only (DL) traffic; (2) timing information only (packet arrival timestamps); and (3) packet size information only. Table 3 presents the accuracy of various classifiers under these conditions.

Table 2: Parameters of each classifier.

Model	Main Parameters / Architecture
KNN	$k = 5$; weight: distance; distance metric: L_2 (Euclidean)
SVM	$C = 4$; kernel: RBF; $\gamma = 0.125$
MLP	Batch size: 64; 4 fully-connected layers with hidden dimensions 512, 256, 128, and ReLU activations
CNN	Two convolutional layers (10 and 20 filters, kernel size 10×10, stride 5), each followed by ReLU and max-pooling; dropout (0.25 and 0.5); final dense layers: 64-ReLU and softmax

Table 3: Classification accuracy (%) of different models using various feature subsets.

Model	Full	UL	DL	UL+DL T	UL+DL S	DL T	DL S
CNN	99.02	98.86	97.55	71.76	98.84	57.18	97.71
SVM	95.65	85.88	81.89	78.45	87.82	59.39	68.13
KNN	95.81	93.87	90.35	91.58	93.63	82.84	85.29
MLP	97.59	96.44	90.75	89.52	94.23	73.78	79.00

Note: **Full** uses all features; **UL**/**DL** refer to uplink/downlink only; **UL+DL T/S** combine timing/size features from both directions; **DL T/S** use only downlink timing/size.

To remove the packet size or time feature from our 2D representation, we apply simple transformations: for timing-only input, all packet sizes are set to zero; for size-only input, inter-arrival times are set to a constant value across all app samples. Our CNN classifier maintains high accuracy even under constrained conditions, achieving 98.86% and 97.55% accuracy with only uplink and downlink data respectively. When limited to packet-size information, the CNN accuracy remains robust at 98.84% (combined UL and DL sizes) and 97.71% (DL sizes only). However, the performance drops with only timing information, primarily because zeroing packet sizes collapses the input to a 1-D representation, which is less suitable for our CNN architectures.

In comparison, traditional classifiers such as KNN and MLP also demonstrate resilience to feature restrictions. Notably, even the simplest SVM model, with access only to limited directional and statistical features, achieves 59% accuracy when classifying 49 applications, further illustrating the inherent vulnerability of encrypted traffic to fingerprinting attacks.

4.2.5 Open-World Scenario Analysis. We evaluate the open-world setting where the adversary has access only to a limited subset of *known* application traffic, while traffic from other *unseen* applications remains available. The attacker's objective is to detect if a specific target application has been used by the victim.

For this scenario, we divide the application traffic into two subsets. The first subset consists of N *known* application activities, used to train the classifier. The remaining activities are treated as *unseen*

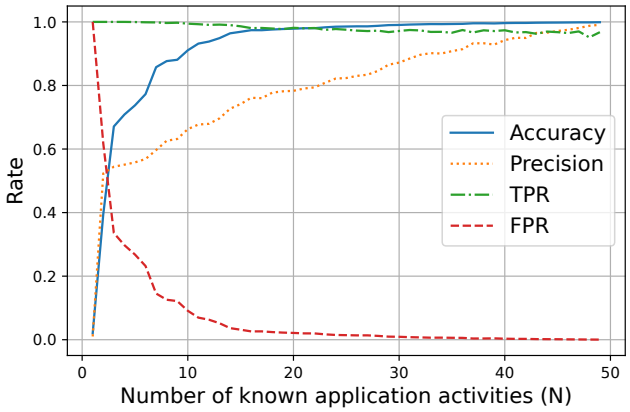


Figure 3: Impact of increasing the number of known application activities N on classification performance in the open-world setting.

and excluded from training. We measure the performance of our attack in function of the different number of known applications N. Specifically, we train the model using only data from known apps and subsequently test our attack on traffic from all apps, both from the known and from the unknown sets. We assess the detection performance in this setting using four metrics: accuracy, precision, true positive rate (TPR), and false positive rate (FPR). The results in fig. 3 show how these metrics vary as the size of the known application set increases.

A higher number of known applications can improve the model's ability to detect targeted traffic. The true positive rate (TPR) remains consistently above 95%, indicating that the attacker can reliably identify the presence of the target application once it occurs. When the known application set includes 15 out of 49 total applications, which is approximately 30% of the total, the model achieves high accuracy exceeding 95% and maintains a low false positive rate (FPR) below 5%.

Precision is initially lower due to the fact that traffic from unseen applications can exhibit statistical patterns similar to those of known apps, leading to false positives. However, precision improves as more known applications are added to the training set. To further improve precision, an attacker can selectively include applications that are functionally similar to the target app in the known set. For example, if the target is a messaging application, incorporating traffic from other messaging apps as non-target classes in the known set can help the model better learn discriminative features and reduce false positives.

4.2.6 App Software Version. In order to evaluate the effects of smartphone app software versions on the classifier's accuracy, we first train our model using traffic from a single version of each application. We then collect an attack dataset consisting of four activities performed using 3 to 6 different versions of three apps. The results are shown in table 4.

In most cases, the model achieves near-perfect recognition for newer versions, while a few older versions, specifically, YouTube v16.36.35 and Instagram v199.0.0.34.119, show noticeable drops

Table 4: Classification accuracy (%) for app activities on different versions over a 12-month period.

App & Activity		Version	Acc.
YouTube		17.48.44	100.00
		17.46.37	100.00
		17.40.41	100.00
		17.26.35	100.00
		17.01.36	100.00
		16.36.35	0.00
WhatsApp	Text chat	2.22.24.78	91.89
	Video call		100
	Text chat	2.22.23.77	97.29
	Video call		100
	Text chat	2.22.20.80	86.49
	Video call		75.00
Instagram		262.0.0.24.327	100.00
		254.0.0.19.109	100.00
		242.0.0.16.111	100.00
		217.0.0.15.474	100.00
		199.0.0.34.119	46.67

in accuracy. This suggests that the classifier adapts well to traffic from similar app versions but may struggle when an app undergoes substantial changes in its data flow or internal architecture. For example, with YouTube, manual examinations revealed that the poor classification performance for the older version is due to a protocol difference, where the older version preferred GQUIC over QUIC for delivering video content. Consequently, we believe that an attacker does not need to update the attack model frequently; updates are only necessary when a major update substantially alters the app's traffic characteristics. In our experiments, the high classification accuracy for YouTube persisted for ten months.

4.2.7 Background Activity Detection. Background activity, also known as the IDLE state, happens when the UE is connected to the e/gNB but no active user activities are performed. This state is periodically triggered when the UE connects to the network to check for notifications, such as FCM or APN. We simulate this scenario by connecting a victim UE to a network and then keeping several apps, including video, social, and communication services running in the background, while performing no additional actions on the UE. Our results are shown in fig. 4. Our classifier is able to correctly identify the background states in 98% of the cases, and the IDLE state identification has a 92% precision and a 95% F1-score.

4.2.8 Real-World Attack Validation. To validate the feasibility of our attack in a real network environment, we connect our UEs to a commercial LTE network, and a MITM Relay is deployed to monitor the traffic exchanged between the UE and the network. Our classification model is trained solely on traffic collected from our private network. We evaluate our attack by collecting traffic from the commercial network; specifically, we collect data from four applications (covering six activities), operating each one for 10 minutes on the commercial network, and then use our pre-trained CNN model to identify the application traffic.

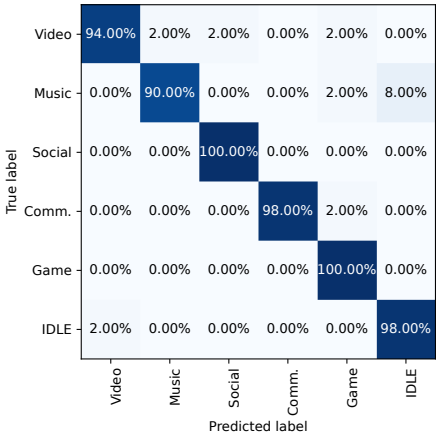


Figure 4: Confusion matrix for IDLE state. IDLE state is correctly identified in 98% of the cases (i.e., recall).

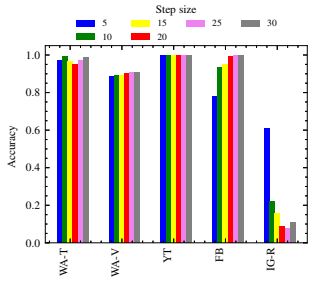


Figure 5: Impact of step size on classification accuracy. Here, WA-T, WA-V, YT, FB, and IG-R are WhatsApp text chat, WhatsApp video call, YouTube, Facebook, and Instagram reels.

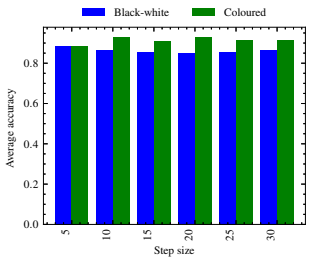


Figure 6: Average accuracy for five activities. Distributions which include directional information (red) perform better than ones without (black-white) at all step sizes.

The evaluation results are shown in table 5. Due to automation challenges on iPhone devices, we had to limit our data collection on this platform to YouTube and Facebook. We observe that the model performs worse on the Samsung S8 than on the Pixel 5, which we attribute to device model differences. For example, Facebook uses the TCP protocol, which is affected by the TCP Maximum Segment Size (MSS). Different smartphone models yield varying MSS values (see table A1 in the Appendix). When connected to the same network, the Pixel 5 sets the TCP MSS at 1360 bytes, while the S8 uses 1260 bytes. Consequently, downlink packets are concentrated around a MSS value of 1400 for the Pixel 5 and 1300 for the S8. We also observed that all tested Android UEs, except the Samsung S8, use a MSS value of 1360, and that some iPhone versions use 1400 while newer ones use 1410. These findings suggest that attackers do not need to train on every smartphone model; a limited set of representative devices suffices. Although device variation can affect accuracy, a well-resourced attacker can generalize by collecting traffic from diverse configurations.

4.2.9 5G-SA Specifics. The 5G specification adds new security features like IMSI encryption, initial Non-Access-Stratum (NAS) message protection, and optional user plane (UP) traffic integrity protection design to protect against data tampering. Additionally, a new Service Data Adaptation Protocol (SDAP) layer is added on top of the PDCP layer to improve the Quality of Service (QoS) functionality by mapping QoS flows to and from Data Radio Bearers (DRBs) at the PDCP sub-layer in both directions. Unfortunately, these changes do not protect against our attacks, as they are passive and remain undetectable by integrity based protection techniques. Furthermore, the PDCP layer remains similar to that of LTE.

To demonstrate this, we test our attack on a self-deployed 5G-SA network, by running YouTube on the iPhone 12. Then, we use the same CNN model which is trained on the private LTE network to identify the app activities. The results show that the CNN model achieves a 66.67% accuracy, which indicates that the PDCP layer in 5G is as susceptible as LTE to our attacks.

Table 5: Accuracy (%) of app activity identification on commercial networks.

App & Activity		Proto.	Samsung S8	Pixel 5	iPhone 12
YouTube		QUIC	71.11	70.00	97.00
Facebook		QUIC	63.64	100.00	100.00
WhatsApp	Text chat	TCP	13.70	56.60	-
	Video call	UDP	84.09	60.00	-
Line	Text chat	TCP	22.83	73.68	-
	Video call	UDP	52.38	100.00	-

4.2.10 Fine-Grained Activity Classification. In our attacker model the adversary does not have access to network-layer information. As such, distinguishing between in-app activities is a challenging task. In particular, the attacker cannot precisely identify when an application session begins, as they lack visibility into IP layer metadata such as packet destinations.

To address this, we propose a recursive fingerprinting approach that operates in a coarse-to-fine manner. Specifically, once the attacker successfully identifies the application in use (e.g., WhatsApp), they can reapply the fingerprinting process with a finer-grained window to infer more specific user activities. We demonstrate this recursive method in our evaluation using WhatsApp-generated traffic. First, we detect sessions associated with WhatsApp, and then we reapply our model to classify in-app tasks such as specific notification events and state information about messages (e.g., sent, delivered, read) based on previously labeled traffic.

As certain activities tend to be associated with specific types of packets (e.g., DNS or fixed size TCP exchanges), our classifier is able to reliably identify TCP packets corresponding to specific events based on their characteristic sizes (e.g., 62 or 69 bytes), thereby validating and extending the findings from [37].

4.2.11 Network Connection Quality. Our attack uses a cellular-relay node to capture traffic at the PDCP layer. As such, while the network environment does have some impact on the traffic's behavior (i.e, traffic shape [6]), the adversary can control the quality of

the connection by adjusting the transmission power or by increasing the bandwidth of the e/gNB component of the relay (which is facing the victim). In order to control the quality of the radio connection segment between the relay and the commercial network the adversary can use radio-frequency power amplifiers in order to improve the received radio power, or use directional antennas to increase the received power and reduce interference from unwanted sources. Alternative setups, such as separating the UE and e/gNB components of the relay and using an out-of-band method to communicate between them, might also be viable options, however, the efficiency of these falls outside the scope of this paper. In our tests, the network environment had minor effects on our attacks.

5 Defense and Evaluation

In this section, we present our defense framework designed to protect against traffic fingerprinting in cellular networks. We first describe the underlying design principles, followed by an overview of the implementation and the latency-overhead trade-off model. Finally, we evaluate the defense using both analytical modeling and experimental measurements.

5.1 Defense Design Principles

Our defense is built upon the following three core principles:

(1) **Strong security foundations.** For our cellular traffic defense framework we take inspiration from Tamaraw [9], a scheme proven secure under an information theoretic framework of ϵ-security. In our framework, the adversary's probability to correctly identify the targeted application is bounded by a tunable parameter $\epsilon \in (0, 1]$, which means that even an optimal adversary cannot achieve a success rate higher than ϵ. We achieve this by clustering n applications into $k = n \cdot \epsilon$ groups and applying uniform traffic shaping within each group, ensuring that any two applications' traffic within the same cluster is indistinguishable to the attacker.

(2) **Feature obfuscation via regularization.** Our defense reshapes both uplink and downlink traffic into fixed-size packets sent at fixed intervals to achieve indistinguishability. This approach not only hides burst patterns and flow rate variations, but also conceals the correlation between uplink and downlink data streams, thus preventing adversaries from exploiting low-level features observed at the PDCP layer.

(3) **Latency and overhead-aware deployment.** Finally, we explicitly model the trade-off between the introduced latency and the shaping rate to achieve practical usability. For each application, we determine the minimal shaping parameters $(\theta^{UL}, \theta^{DL})$ that satisfy its latency constraints, thus minimizing overhead while preserving the desired level of security. This enables efficient grouping of heterogeneous applications under a common defense strategy.

Based on the principles above, we implement and experimentally test our scheme at the PDCP layer of the 4G and 5G protocol stacks, and show that our solution can be transparently deployed within the radio access network without impacting the core network load. This makes our scheme particularly suitable for privacy-sensitive enterprises or private mobile networks.

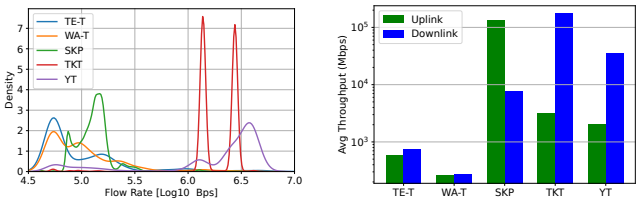


Figure 7: Traffic characteristics of five mobile applications: TE-T (Telegram chat), WA-T (WhatsApp chat), SKP (Skype video), TKT (TikTok), and YT (YouTube). Left: PDF output for instantaneous throughput estimated using KDE, the x-axis represents flow rate in log10 of Bps. Right: Average uplink and downlink throughput (log-scale, Mbps). Text Chat apps show lower and balanced traffic, while video streaming apps exhibit significantly higher downlink rates. SKP (Skype) displays a notable uplink bias.

5.2 Challenges in App Fingerprinting Defense

While the proposed scheme provides strong security guarantees, traffic regularization can introduce non-negligible latency and bandwidth overheads. A key challenge arises from the heterogeneity of mobile application behaviors, as shown in section 5.3. For example, when comparing a video streaming service (e.g., YouTube) and real-time communication applications (e.g., Skype) they both could exhibit similar throughput, but their latency tolerances will differ significantly, primarily due to the buffering mechanisms employed by the video streaming service. Video streaming services can tolerate higher delays, whereas real-time communications require low latencies to maintain interactive performance.

To address this challenge, we introduce a latency and overhead trade-off model that relates the defense configurations (e.g., shaping rates) to the application's delay tolerance. Given an application's traffic profile and performance requirements, we derive the minimal output rate needed to satisfy its delay constraints. This enables us to form application clusters that not only meet the fixed security level ϵ, but also minimize the shaping overhead, facilitating efficient deployment in diverse mobile environments.

5.3 Traffic Patterns in Mobile Applications

Due to the diverse nature of mobile applications, their traffic patterns vary significantly in throughput, burstiness, and directional balance. In fig. 7 we show the traffic characteristics of five representative applications. The left side of the figure shows the estimated probability density function (PDF) outputs for instantaneous throughput, computed from average throughput measured over short time intervals using kernel density estimation (KDE) [41]. The x-axis represents the instantaneous flow rate (in Bps, on a logarithmic scale), while the y-axis denotes the estimated probability density. To emphasize the differences between applications, zero throughput intervals were excluded from the density estimates. Across all applications, these zero-rate intervals constitute a significant portion of the data, reflecting the bursty nature of the traffic; that is, traffic tends to arrive in sudden large bursts, followed by periods of silence.

As shown, traffic rate distributions vary significantly between applications. For instance, Skype peaks around 10^5 Bps (Bytes per second), indicating frequent low-rate bursts, while TikTok peaks near 10^6 Bps with much higher bursts. The right side of fig. 7 compares the average uplink and downlink throughput across these applications. Text chat apps have much lower overall throughput than video apps, with relatively balanced traffic. In contrast, video streaming apps produce far more downlink traffic than uplink traffic. Notably, Skype displays an inverse pattern, with higher throughput observed in the uplink direction.

5.4 Regularization-Based Defense Design

We now describe the architecture and operation of our proposed regularization-based defense. The defense mechanism is implemented at the PDCP layer on both the UE and base station (gNB) side, without modifying the application layer or the core network stack. This ensures transparency to the user and application.

Briefly, according to 3GPP specifications[2, 3], there are two PDCP entity types based on data flows: transmitting entities and receiving entities. Transmitting entities get service data units (SDUs) from upper layers (e.g., IP, SDAP, RRC) and construct PDCP packets by adding sequence numbers, header compression, and ciphering before submitting them to the lower RLC layer. The receiving entities perform the process in reverse to recover the original payload.

5.4.1 Transmitting Entity. We introduce a traffic shaper that buffers application traffic and releases fixed-size payloads at regular intervals. This module is activated at the start of an application session and remains active until the session ends. Upon activation, it retrieves its shaping parameters, specifically, the fixed packet size, denoted by s, and the packet transmission interval ρ, from a precomputed configuration or from the upper layer (SDAP), which is derived from the app's assigned traffic cluster. Once initialized, the module starts a timer that triggers every ρ milliseconds. Incoming SDUs from upper layers (e.g., IP or SDAP) are first stored in a FIFO queue Q_{snd}, and wait until next scheduled transmission. On each timer event, s bytes are extracted from Q_{snd} to construct a PDCP SDU. If the queue contains fewer than s bytes at timer expiration, padding is added, and if no data is available, a dummy packet consisting of padding only is transmitted. This ensures that traffic is emitted with uniform patterns, regardless of application behavior.

5.4.2 Receiving Entity. To properly reassemble the original IP packets, the receiving entity must identify the start and end of each IP packet and distinguish actual payload data from padding. In our design, we use a secondary FIFO queue Q_{rcv}, and a custom reassembly function which is executed after the standard PDCP procedure. As such, after the PDCP decryption and integrity checks are concluded, the receiver checks if the PDCP payload contains a valid header. If not, and the receiver is not waiting for data, the packet is treated as a dummy and discarded. Valid payloads are appended to Q_{rcv}. Since the first IP packet is always contained in the first PDCP packet of the bearer, the receiving entity can directly extract the expected IP packet size $|ip|$ from the header of this first PDCP packet. After appending the unencrypted PDCP packet payload to Q_{rcv}, the receiver updates the queue length $|Q_{rcv}|$ accordingly and subsequently performs the following:

(1) If $|ip| > |Q_{rcv}|$, the receiver waits for additional PDCP packets to complete the reassembly.
(2) If $|ip| \leq |Q_{rcv}|$, an IP packet of size $|ip|$ is extracted from Q_{rcv}, and a IP checksum validation is performed.

Any remaining data in the queue is checked for `null` bytes, which indicate padding. These are discarded before processing the next IP packet.

5.4.3 Direction Binding. Each application generates two traffic flows: uplink and downlink. These two flows can be shaped separately. However, treating them as independent may leak information through cross-directional correlation. To prevent this, we bind each pair of shaping parameters (s^{UL}, ρ^{UL}) and (s^{DL}, ρ^{DL}) as a joint configuration assigned per application cluster, where s^{UL} and s^{DL} denote the fixed packet sizes for uplink and downlink, and ρ^{UL} and ρ^{DL} represent their respective transmission intervals. This ensures a consistent and uninformative traffic pattern.

5.5 Latency and Overhead Trade-Off Model

Any regularization-based defense inevitably introduces transmission delays, as arriving packets may need to wait until their scheduled send occasion. To quantify and control this delay, we present a latency model that estimates the expected per-packet delay L as the sum of two components:

(1) **Scheduling delay**. The time a packet waits for the next transmission occasion, which is bounded by the fixed packet transmission interval ρ.
(2) **Queuing delay**. The additional delay is caused when the input traffic rate exceeds the shaping rate, resulting in the traffic data being split and sent in multiple intervals.

We define the shaping rate as $\theta = \frac{s}{\rho}$, where s is the fixed packet size and ρ is the fixed transmission interval. We also denote λ as a random variable representing the instantaneous input traffic rate, with $P(\lambda = x_i)$ as its empirical probability mass function, derived from observed traffic traces. Using this notation, the latency experienced by a packet, given an input rate λ, is defined as:

$$L(\lambda) = \begin{cases} \rho, & \lambda \leq \theta, \\ \lfloor \frac{\lambda}{\theta} \rfloor \cdot \rho + \rho, & \lambda > \theta, \end{cases} \tag{1}$$

where $\lfloor x \rfloor$ denotes the floor function, which rounds x down to the nearest integer. This function is used to capture the number of complete transmission intervals required to clear the accumulated data when the input rate λ exceeds the shaping rate θ.

Specifically:

(1) When $\lambda \leq \theta$, the traffic shaper's capacity is *sufficient* to handle the incoming traffic immediately, so each packet experiences only the scheduling delay of ρ.
(2) When $\lambda > \theta$, the shaper's capacity is *insufficient*. In this case, the ratio $\frac{\lambda}{\theta}$ represents the number of intervals needed to transmit the incoming data. The floor function $\lfloor \frac{\lambda}{\theta} \rfloor$ gives the count of complete intervals required. An additional delay of ρ is then added to account for the scheduling delay in the next interval. Together, $\lfloor \frac{\lambda}{\theta} \rfloor \cdot \rho + \rho$ models the total delay, reflecting both the time needed to transmit the queued data and the extra waiting time for the next transmission slot.

The latency overhead is computed as the expectation:

$$E[L] = \sum_{x_i} L(x_i) \cdot P(\lambda = x_i). \tag{2}$$

In our experiments, we determine the optimal shaping rate θ for traffic flows by conducting a grid search over a predefined list of candidate values for both θ and the interval ρ. Specifically, we generate candidate configurations by selecting various (s, ρ) pairs (with $\theta = \frac{s}{\rho}$) and computing the corresponding expected latency $E[L]$ using Equation (2). We then choose the configuration that minimizes the overhead while ensuring that $E[L]$ remains below a desired latency threshold. Alternatively, optimization-based approaches can be employed; however, the grid search method proves straightforward and effective for our use case.

5.6 Application Clustering

Once we determine each application's uplink and downlink shaping rates according to its latency tolerance, we perform clustering to group similar applications into classes.

As stated, for an application a, we obtain two shaping rates: θ_a^{UL}, the rate corresponding to the uplink traffic of application a; and θ_a^{DL} the rate for downlink, both computed from the model in section 5.5. We then apply k-means clustering over the $(\theta_a^{UL}, \theta_a^{DL})$ pairs to assign the applications into k clusters.

Within each cluster, the configuration of the traffic shaper is based on the maximum uplink traffic as well as the downlink traffic within the cluster:

$$(\theta_C^{UL}, \theta_C^{DL}) = \left(\max_{a \in C} \theta_a^{UL}, \max_{a \in C} \theta_a^{DL} \right). \tag{3}$$

This ensures that all applications in the same cluster can be safely reshaped using the same parameters without exceeding their individual delay bounds. We define the cost of the defense as the ratio of the total throughput after shaping to the total throughput before shaping, which is expressed as:

$$Cost = \frac{\sum_{a \in \mathcal{A}} (\theta_{c(a)}^{UL} + \theta_{c(a)}^{DL})}{\sum_{a \in \mathcal{A}} (\Lambda_a^{UL} + \Lambda_a^{DL})}, \tag{4}$$

where $\mathcal{A}$ is the set of all applications, $C(a)$ denotes the cluster to which application a belongs, and Λ_a^{UL}, Λ_a^{DL} represent the original uplink and downlink traffic throughput of application a.

Finally, each application cluster is assigned a fixed pair of shaping parameters (s^{UL}, ρ^{UL}) and (s^{DL}, ρ^{DL}), derived from the selected θ values. These parameters are applied by the traffic shaper to all flows in the same cluster, ensuring uniform traffic shaping. The selection of (s, ρ) for each direction can be further tailored to the cluster traffic characteristics. A smaller ρ results in lower transmission latency for small packets, while a larger ρ helps avoid excessive segmentation by allowing larger packets to be sent as a whole. This trade-off enables more efficient and application-aware shaping within each cluster.

5.7 Defense Evaluation

5.7.1 Analytical Evaluation. To evaluate our defense mechanism, we examine how our proposed scheme balances bandwidth overhead, latency overhead, and security levels. First, we derive the instantaneous throughput distributions for both uplink and downlink traffic for all 49 applications (as in fig. 7). Then, using our latency model from section 5.5, we translate each application latency requirement into a specific defense configuration.

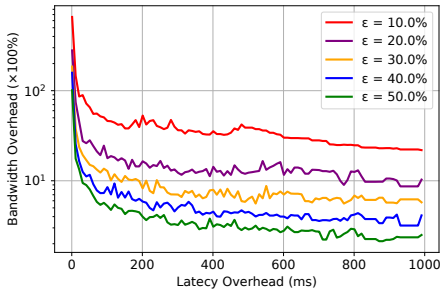


Figure 8: Bandwidth overhead vs. latency overhead at different security levels. Lower latency configurations increase the bandwidth overhead. The graph highlights the trade-off between latency and bandwidth.

We use a non-uniform ϵ-security definition (commonly used in this setting [9]), meaning that the adversary's *average* accuracy across all applications is bounded by ϵ, but individual application accuracies are not. In practice, we cluster n total applications into $n \times \epsilon$ groups and unify their throughput shaping parameters to obscure traffic patterns. The bandwidth overhead introduced by our defense arises from aligning each application's traffic to the maximum uplink and downlink defense throughput within its assigned group (see eq. (4)). The results are shown in fig. 8 where we plot the bandwidth overheads (y-axis) against different latency constraints (x-axis) for various security levels $\epsilon \in \{10\%, 20\%, 30\%, 40\%, 50\%\}$. These translate to the following:

(1) **Security is proportional to the bandwidth overhead.** As shown in fig. 8, the curve for $\epsilon = 10\%$ (i.e., a stronger security requirement on the adversary's accuracy) stays above those for higher ϵ values at all latency settings.
(2) **Bandwidth overheads can be reduced by adjusting the latency tolerance.** Across all curves, increasing the latency tolerance from near-zero to around 1000 ms substantially lowers the bandwidth overhead. This suggests that if an application (or user) can tolerate longer delays, the defense can efficiently shape traffic without generating excessive dummy packets or padding.
(3) **Overheads are lower bounded.** At each security level, under very low-latency requirements, even a slight relaxation dramatically reduces overhead. This indicates that under strict delay constraints, the latency requirement is the primary factor contributing to the overhead. However, as the permitted latency increases, each curve tends to stabilize and no longer exhibits significant decline, at which point ϵ becomes the primary factor affecting overhead. This behavior corroborates the existence of a theoretical lower bound to the bandwidth overhead, as demonstrated by Cai et al. [9].

Table 6: Throughput and latency of the proposed defense under varying packet transmission interval ρ.

Transmission interval ρ (ms)	10	9	8	7	6	5	4	3	2	1	Baseline
Measured throughput (Mbits/s)	1.15	1.28	1.44	1.63	1.92	2.30	2.88	3.82	5.74	11.30	25.8
Shaped throughput (Mbits/s)	1.20	1.33	1.50	1.71	2.00	2.40	3.00	4.00	6.00	12.00	–
Latency (ms)	33.915	31.457	29.000	28.873	27.011	26.871	25.081	25.065	23.520	21.697	27.575

Note: Measurements are obtained using srsRAN_4G with a srsUE/srsENB testbed (USRP B210, SISO) on an AMD Ryzen 9 7950X CPU with 64GB RAM.

5.7.2 Defense Performance. In addition to the analytical evaluation, we also implement our defense in a LTE testbed to assess its real-world performance. Specifically, we implemented the algorithm described in section 5.4 using the open-source project srsRAN_4G. The modified srsUE and the srsENB are connected via two USRP B210, with the srsENB cell configured to use 100 physical resource blocks in single-input single-output (SISO) mode. We simplify the configuration by fixing the packet size s to 1500 bytes and varying the transmission interval ρ to control the shaping throughput $\theta = \frac{s}{\rho}$. We then measure the effective throughput using iPerf and evaluate end-to-end latency by generating ICMP traffic and recording the round-trip time. For comparison, we also use the unmodified srsUE and srsENB as the baseline.

As table 6 shows, the measured throughput closely follows the shaped values across all tested ρ, with minor degradation of typically less than 5% due to dummy packets and padding. Without defense, the baseline throughput reaches 25.8Mbits/s. The maximum throughput achieved under our defense is 11.3 Mbits/s. This value is close to the theoretical shaping limit of 12 Mbits/s, which is imposed by our implementation: the minimum packet sending interval is 1 ms, and with $s = 1500$ bytes this yields a shaping rate of 12 Mbits/s. Regarding latency, we observe that when the shaping interval ρ is small, the additional delay introduced by the defense is negligible. Latency increases when ρ becomes larger, as packets experience additional queuing before transmission. Interestingly, reducing the packet sending interval yields even lower latency than the baseline, due to the finer granularity of packet scheduling. From the result, we can learn that when the defense is correctly configured, it does not impact the operation of the applications.

6 Conclusion

Our traffic fingerprinting attack underscores a substantial and realistic privacy risk within current cellular networks, largely due to its high accuracy and demonstrated practicality across diverse real-world scenarios. While adversaries require certain hardware and software tools to intercept the radio link [16, 27, 33], these remain relatively easy to obtain, enabling potential exploitation by both cybercriminals and large organizations involved in espionage. Such targeted attacks can severely threaten individual privacy and security by revealing fine-grained user activities, from specific application usage to browsing behaviors and messaging interactions, and thereby facilitating a wide array of malicious operations, including phishing, financial fraud, identity theft, unauthorized profiling, targeted advertising, surveillance, and censorship. As cellular networks continue to expand in coverage and user base, these risks demand proactive attention from network operators, application developers, and policymakers.

While our proposed regularization-based defense effectively mitigates fingerprinting threats by providing robust security guarantees, it introduces unavoidable performance trade-offs, particularly in terms of bandwidth overhead and latency. Despite its backward compatibility and flexibility, the overhead may hinder widespread adoption in environments with strict latency constraints or limited resources. Future research should focus on optimizing this trade-off, potentially through adaptive defense strategies or more sophisticated traffic shaping techniques to maintain strong security without significant performance penalties.

Acknowledgments

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. We are grateful to the anonymous reviewers and to Marius Muench for their insightful feedback, which helped improve the paper. Computations were performed using the University of Birmingham's BEAR Cloud, which provides flexible resources for intensive research tasks.

References

[1] 2022. Free Android Market Data, History, Rankings | since 2011. https://www.androidrank.org/ [Online; accessed 05-September-2022].

[2] 3GPP. 2008. *Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification.* Technical Specification TS 36.323. 3rd Generation Partnership Project (3GPP). https://www.3gpp.org/ftp/Specs/archive/36_series/36.323/ Release 8, under change control.

[3] 3GPP. 2018. *NR; Packet Data Convergence Protocol (PDCP) specification.* Technical Specification TS 38.323. 3rd Generation Partnership Project (3GPP). https://www.3gpp.org/ftp/Specs/archive/38_series/38.323/ Release 15, under change control.

[4] 3GPP. 2023. *Study on 5G security enhancements against False Base Stations (FBS).* Technical Report TR 33.809. 3rd Generation Partnership Project (3GPP). https://www.3gpp.org/ftp/Specs/archive/33_series/33.809/ Release 18, under change control.

[5] Ahmed Abusnaina, Rhongho Jang, Aminollah Khormali, DaeHun Nyang, and David Mohaisen. 2020. DFD: Adversarial Learning-based Approach to Defend Against Website Fingerprinting. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications.* 2459–2468. https://doi.org/10.1109/INFOCOM41043.2020.9155465

[6] Sangwook Bae, Mincheol Son, Dongkwan Kim, CheolJun Park, Jiho Lee, Sooel Son, and Yongdae Kim. 2022. Watching the Watchers: Practical Video Identification Attack in LTE Networks. In *31st USENIX Security Symposium (USENIX Security 22).* USENIX Association, Boston, MA, 1307–1324.

[7] Jaejong Baek, Pradeep Kumar Duraisamy Soundrapandian, Sukwha Kyung, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. 2023. Targeted Privacy Attacks by Fingerprinting Mobile Apps in LTE Radio Layer. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).* 261–273. https://doi.org/10.1109/DSN58367.2023.00035

[8] Nicola Bui and Joerg Widmer. 2016. OWL: a Reliable Online Watcher for LTE Control Channel Measurements. In *ACM All Things Cellular (MobiCom Workshop)* (New York, USA).

[9] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14).* Association for Computing Machinery, New York, NY, USA, 227–238. https://doi.org/10.1145/2660267.2660362

[10] Zishuai Cheng, Mihai Ordean, Flavio D. Garcia, Baojiang Cui, and Dominik Rys. 2023. Watching your call: Breaking VoLTE privacy in LTE/5G networks. *Proc. Priv. Enhancing Technol.* 2023, 2 (2023). https://doi.org/10.48550/ARXIV.2301.02487

[11] Cybersecurity and Infrastructure Security Agency. 2020. Best Practices for Using Public Wi-Fi: Tip Card. https://www.cisa.gov/sites/default/files/publications/Best%20Practices%20for%20Using%20Public%20WiFi.pdf [Online; accessed 14-April-2025].

[12] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *2012 IEEE Symposium on Security and Privacy*. 332–346. https://doi.org/10.1109/SP.2012.28

[13] Robert Falkenberg and Christian Wietfeld. 2019. FALCON: An Accurate Real-time Monitor for Client-based Mobile Network Data Analytics. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Waikoloa, Hawaii, USA. https://doi.org/10.1109/GLOBECOM38437.2019.9014096 arXiv:1907.10110

[14] Jiajun Gong and Tao Wang. 2020. Zero-delay Lightweight Defenses against Website Fingerprinting. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 717–734. https://www.usenix.org/conference/usenixsecurity20/presentation/gong

[15] Gaofeng He, Ming Yang, Junzhou Luo, and Xiaodan Gu. 2015. A novel application classification attack against Tor. *Concurrency and Computation: Practice and Experience* 27, 18 (2015), 5640–5661. https://doi.org/10.1002/cpe.3593 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3593

[16] Tuan Dinh Hoang, CheolJun Park, Mincheol Son, Taekkyung Oh, Sangwook Bae, Junho Ahn, BeomSeok Oh, and Yongdae Kim. 2023. LTESniffer: An Open-source LTE Downlink/Uplink Eavesdropper. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Guildford, United Kingdom) *(WiSec '23)*. Association for Computing Machinery, New York, NY, USA, 43–48. https://doi.org/10.1145/3558482.3590196

[17] James K Holland and Nicholas Hopper. 2020. Regulator: A straightforward website fingerprinting defense. *arXiv preprint arXiv:2012.06609* (2020).

[18] Chengshang Hou, Gaopeng Gou, Junzheng Shi, Peipei Fu, and Gang Xiong. 2020. WF-GAN: Fighting Back Against Website Fingerprinting Attack Using Adversarial Learning. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–7. https://doi.org/10.1109/ISCC50000.2020.9219593

[19] Syed Rafiul Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. 2018. LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE. In *Network and Distributed System Security Symposium*. https://api.semanticscholar.org/CorpusID:3387805

[20] Md Ruman Islam, Raja Hasnain Anwar, Spyridon Mastorakis, and Muhammad Taqi Raza. 2024. Characterizing Encrypted Application Traffic Through Cellular Radio Interface Protocol. In *2024 IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*. 321–329. https://doi.org/10.1109/MASS62177.2024.00050

[21] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an Efficient Website Fingerprinting Defense. In *Computer Security – ESORICS 2016*, Ioannis Askoxylakis, Sotiris Ioannidis, Sokratis Katsikas, and Catherine Meadows (Eds.). Springer International Publishing, Cham, 27–46.

[22] Katharina Kohls, David Rupprecht, Thorsten Holz, and Christina Pöpper. 2019. Lost Traffic Encryption: Fingerprinting LTE/4G Traffic on Layer Two. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks* (Miami, Florida) *(WiSec '19)*. Association for Computing Machinery, New York, NY, USA, 249–260. https://doi.org/10.1145/3317549.3323416

[23] Nitya Lakshmanan, Abdelhak Bentaleb, Byoungjun Choi, Roger Zimmermann, Jun Han, and Min Suk Kang. 2022. On Privacy Risks of Watching YouTube over Cellular Networks with Carrier Aggregation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 1, Article 19 (March 2022), 22 pages. https://doi.org/10.1145/3517261

[24] Nitya Lakshmanan, Abdelhak Bentaleb, Byoungjun Choi, Roger Zimmermann, Jun Han, and Min Suk Kang. 2022. On Privacy Risks of Watching YouTube over Cellular Networks with Carrier Aggregation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 1 (2022), 19:1–19:22. https://doi.org/10.1145/3517261

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[26] Junming Liu, Yanjie Fu, Jingci Ming, Yong Ren, Leilei Sun, and Hui Xiong. 2017. Effective and Real-Time In-App Activity Analysis in Encrypted Internet Traffic Streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17)*. Association for Computing Machinery, New York, NY, USA, 335–344. https://doi.org/10.1145/3097983.3098049

[27] N. Ludant, P. Robyns, and G. Noubir. 2023. From 5G Sniffing to Harvesting Leakages of Privacy-Preserving Messengers. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 3146–3161. https://doi.org/10.1109/SP46215.2023.10179353

[28] Xiapu Luo, Peng Zhou, Edmond WW Chan, Wenke Lee, Rocky KC Chang, Roberto Perdisci, et al. 2011. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows.. In *NDSS*, Vol. 11.

[29] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. 2021. Defeating DNN-Based Traffic Analysis Systems in Real-Time With Blind Adversarial Perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2705–2722. https://www.usenix.org/conference/usenixsecurity21/presentation/nasr

[30] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society* (Chicago, Illinois, USA) *(WPES '11)*. Association for Computing Machinery, New York, NY, USA, 103–114. https://doi.org/10.1145/2046556.2046570

[31] Tao Qin, Lei Wang, Zhaoli Liu, and Xiaohong Guan. 2015. Robust Application Identification Methods for P2P and VoIP Traffic Classification in Backbone Networks. *Know.-Based Syst.* 82, C (jul 2015), 152–162. https://doi.org/10.1016/j.knosys.2015.03.002

[32] Mohammad Saidur Rahman, Mohsen Imani, Nate Mathews, and Matthew Wright. 2021. Mockingbird: Defending Against Deep-Learning-Based Website Fingerprinting Attacks With Adversarial Traces. *IEEE Transactions on Information Forensics and Security* 16 (2021), 1594–1609. https://doi.org/10.1109/TIFS.2020.3039691

[33] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. 2019. Breaking LTE on layer two. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1121–1136.

[34] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. 2020. IMP4GT: IMPersonation Attacks in 4G NeTworks.. In *NDSS*.

[35] Brendan Saltaformaggio, Hongjun Choi, Kristen Johnson, Yonghwi Kwon, Qi Zhang, Xiangyu Zhang, Dongyan Xu, and John Qian. 2016. Eavesdropping on {Fine-Grained} User Activities Within Smartphone Apps Over Encrypted Network Traffic. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*.

[36] Brendan Saltaformaggio, Hongjun Choi, Kristen Johnson, Yonghwi Kwon, Qi Zhang, Xiangyu Zhang, Dongyan Xu, and John Qian. 2016. Eavesdropping on Fine-Grained User Activities within Smartphone Apps over Encrypted Network Traffic. In *Proceedings of the 10th USENIX Conference on Offensive Technologies* (Austin, TX) *(WOOT'16)*. USENIX Association, USA, 69–78.

[37] Theodor Schnitzler, Katharina Kohls, Evangelos Bitsikas, and Christina Pöpper. 2023. Hope of Delivery: Extracting User Locations From Mobile Instant Messengers. In *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/hope-of-delivery-extracting-user-locations-from-mobile-instant-messengers/

[38] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1357–1374. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/schuster

[39] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert. 2015. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. *ArXiv* abs/1510.07563 (2015). https://api.semanticscholar.org/CorpusID:11851582

[40] Tal Shapira and Yuval Shavitt. 2019. FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 680–687. https://doi.org/10.1109/INFCOMW.2019.8845315

[41] George R Terrell and David W Scott. 1992. Variable kernel density estimation. *The Annals of Statistics* (1992), 1236–1265.

[42] Hoang Duy Trinh, Ángel Fernández Gambín, Lorenza Giupponi, Michele Rossi, and Paolo Dini. 2021. Mobile Traffic Classification Through Physical Control Channel Fingerprinting: A Deep Learning Approach. *IEEE Transactions on Network and Service Management* 18, 2 (2021), 1946–1961. https://doi.org/10.1109/TNSM.2020.3028197

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

[44] Tao Wang and Ian Goldberg. 2017. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1375–1390. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao

[45] Charles V Wright, Scott E Coull, and Fabian Monrose. 2009. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis.. In *NDSS*, Vol. 9.

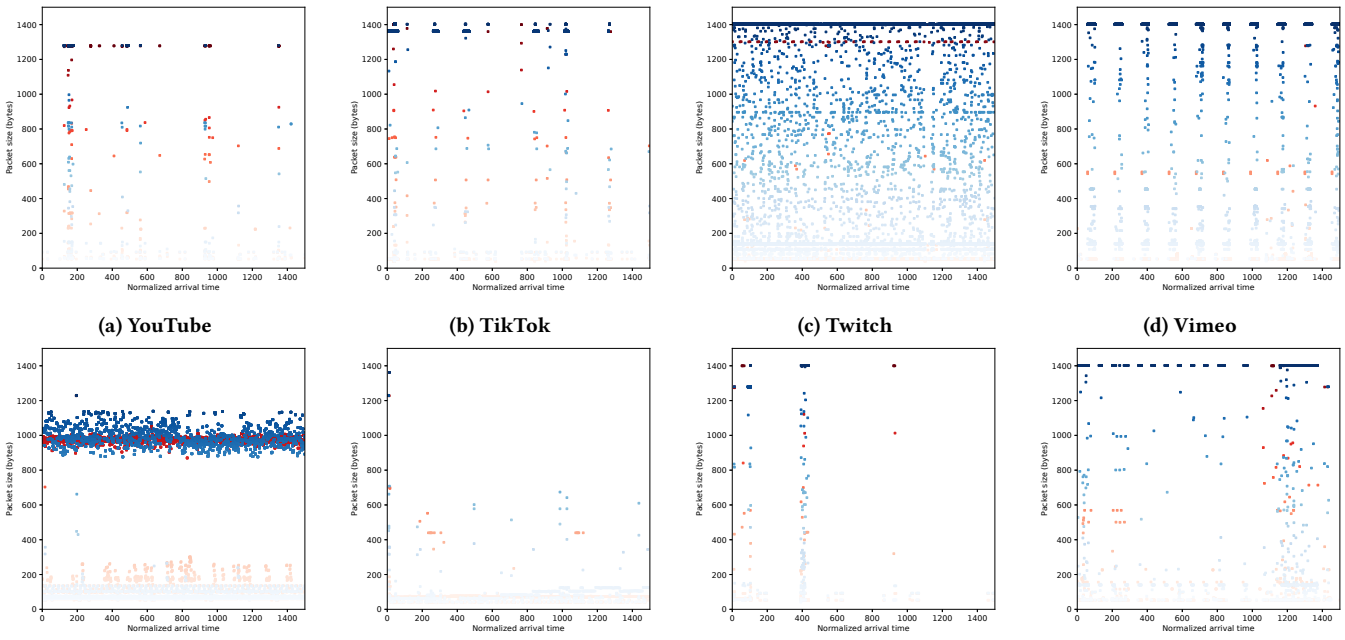
[46] Xi Xiao, Xiang Zhou, Zhenyu Yang, Le Yu, Bin Zhang, Qixu Liu, and Xiapu Luo. 2024. A comprehensive analysis of website fingerprinting defenses on Tor. *Computers & Security* 136 (2024), 103577. https://doi.org/10.1016/j.cose.2023.103577

[47] Liuqun Zhai, Zhuang Qiao, Zhongfang Wang, and Dong Wei. 2021. Identify What You are Doing: Smartphone Apps Fingerprinting on Cellular Network Traffic. In *2021 IEEE Symposium on Computers and Communications (ISCC)*. 1–7. https://doi.org/10.1109/ISCC53001.2021.9631415

Appendix

Table A1: Maximum Segment Size (MSS) values of different UEs when connected to the same carrier.

Phone	OS Version	Chipset	Baseband Version	MSS (bytes)
iPhone X Max	15.4	Apple A12 Bionic	4.03.02	1400
iPhone 11	16.3	Apple A13 Bionic	4.00.00	1400
iPhone 12	15.4.1	Apple A14 Bionic	2.53.01	1410
iPhone 13 Pro Max	16.1.2	Apple A15 Bionic	2.12.02	1410
iPhone 14	16.3	Apple A15 Bionic	1.41.02	1410
iPhone 15	17.2.1	Apple A16 Bionic	–	1410
Samsung S8	9	Exynos 8895	G9500ZHS6DUD1	1260
Google Pixel 5	13	Snapdragon 765G	TQ1A.221205.011	1360
Honor V30	Harmony OS 2.0.0	Kirin 990	21C20B710S000C000	1360
ZTE Axon 30 Ultra 5G	11	Snapdragon 888 5G	MPSS.HI.4.0.c8-00016-LC ALL_PACK-1.422254.58	1360
Huawei Mate 30 5G	10	Kirin 990	21C20B516S000C000	1360



(a) YouTube (b) TikTok (c) Twitch (d) Vimeo

(e) Messenger - video. (f) Garena (game) (g) Spotify (h) Twitter

Figure A1: PSD distributions of app activities. The first row corresponds to video streaming activities on YouTube, TikTok, Twitch, and Vimeo. The second row shows the traffic behavior corresponding to Messenger video call activities, followed by the Garena gaming activities. The last two images represent the behaviors of Spotify music and Twitter. The red and blue pixels represent uplink and downlink traffic, respectively.

Table A2: Classification performance for 40 apps and 49 activities.

Category	App	Activity	Precision	Recall	F1-Score	App Version
Video	YouTube	–	1.00	0.98	0.99	17.17.37
	TikTok	–	1.00	1.00	1.00	27.3.4
	Vimeo	–	1.00	0.94	0.97	3.65.3
	TED	–	1.00	1.00	1.00	7.4.64
	WeTV	–	0.91	0.86	0.89	5.5.8.9820
	Bilibili	–	1.00	1.00	1.00	3.16.0
	Twitch	–	0.93	0.76	0.84	14.3.2
	iQIYI	–	1.00	1.00	1.00	4.11.0
Music	Spotify	–	0.96	1.00	0.98	8.7.92.521
	YouTube Music	–	1.00	0.96	0.98	5.36.51
	SoundCloud	–	1.00	0.98	0.99	2022.12.05-release
	QQ Music	–	1.00	0.98	0.99	12.0.0.9
	Shazam	–	1.00	1.00	1.00	13.9.0-221205
	KuGou	–	1.00	1.00	1.00	11.4.2
	NetEase Cloud Music	–	1.00	1.00	1.00	8.9.0
	Pandora	–	1.00	1.00	1.00	2212.1
Social	Facebook	Reels	1.00	1.00	1.00	396.1.0.28.104
	Instagram	Reels	0.96	1.00	0.98	263.2.0.19.104
		Text	0.92	0.98	0.95	
	Twitter	–	1.00	1.00	1.00	9.69.1-release.0
	Reddit	–	0.94	1.00	0.97	2022.45.0
	Pinterest	–	1.00	1.00	1.00	10.44.0
	Quora	–	1.00	1.00	1.00	3.2.20
	Weibo	–	1.00	1.00	1.00	12.12.2
	Zhihu	–	1.00	1.00	1.00	8.33.0
Communication	WhatsApp	Text	1.00	1.00	1.00	2.22.24.78
		Video	1.00	1.00	1.00	
	Messenger	Text	1.00	1.00	1.00	390.2.0.29.103
		Video	1.00	1.00	1.00	
	Telegram	Text	1.00	1.00	1.00	9.2.2
		Video	1.00	1.00	1.00	
	WeChat	Text	1.00	0.94	0.97	8.0.30
		Video	1.00	1.00	1.00	
	Snapchat	Text	0.91	1.00	0.95	12.12.0.38
		Video	1.00	1.00	1.00	
	Skype	Text	1.00	0.98	0.99	8.92.0.206
		Video	1.00	1.00	1.00	
	QQ	Text	0.98	1.00	0.99	8.2.11
		Video	0.83	1.00	0.91	
	Line	Text	1.00	1.00	1.00	12.21.1
		Video	0.98	1.00	0.99	
Game	Garena Free Fire	–	0.94	1.00	0.97	1.9.4.1
	PUBG Mobile	–	1.00	1.00	1.00	2.3.0
	Arena of Valor	–	1.00	1.00	1.00	1.48.1.2
	FIFA Mobile: FIFA World Cup	–	1.00	1.00	1.00	18.0.02
	Genshin Impact	–	1.00	0.91	0.95	3.3.0_11741873_11806263
	Hearthstone	–	0.92	0.98	0.95	25.0.159202
	League of Legends: Wild Rift	–	1.00	0.94	0.97	3.5.0.6093
	UNO	–	0.98	0.94	0.96	1.10.3448