Hidden Figures: Fingerprinting Mobile Phones through Passive Background App Traffic

Shiva Azizzadeh[1][0000−0003−1330−1921] and Gunes Acar[1][0000−0002−1621−8333]

Radboud University, Nijmegen, The Netherlands
shiva.azizzadeh@ru.nl
gacar@cs.ru.nl

Abstract. Traffic analysis has been used for over two decades to extract sensitive information from network communications. Most existing approaches rely on user-generated activity, such as web browsing, which introduces constraints on the attack scenario and requires assumptions about user behavior. These limitations raise the question of whether similar attacks remain feasible when no user interaction is present. In this work, we propose a passive fingerprinting attack that solely relies on background smartphone network traffic, generated without user involvement. More precisely, we show that communication endpoints extracted from smartphone background traffic contains sufficient information to distinguish user profiles and link device activity across locations. To this end, we collect 400 hours of Wi-Fi traffic representing various user profiles. We identify factors influencing the success of the attack and optimize the attack accordingly. Our results show that an adversary can successfully distinguish individual users with a best-case $F1$ score of 0.89 and accuracy of 0.99 (mean $F1$=0.72, mean accuracy=0.97). We achieve this performance without complex machine learning models or extensive training phases. Instead, we use lightweight distance-based comparisons, which makes the attack practical and efficient. This leads to a straightforward attack setup that enables an adversary to perform user tracking, even in the absence of interaction and despite common privacy protections.

Keywords: Fingerprinting · Traffic analysis · Mobile phones security.

1 Introduction

Mobile phone users expose valuable personal and behavioral data to network eavesdroppers when using online services [2]. Such data can be leveraged for various purposes such as identifying communication patterns in network traffic [20, 60] or detecting malware through application behavior [28]. Notably, even encrypted traffic leaks fine grained information in realistic settings [51, 58]; it can expose user actions and identify applications [61, 30], reveal users' daily activity patterns and service requirements [32], and localize user movement [50]. Devices also remain distinguishable through transmission characteristics [26] and

communication derived "fingerprints" [62, 2] e.g., web sessions, sets of installed applications, or application behavior [9] even in the presence of encryption [56].

These attacks all rely on network traffic triggered by user interaction, and often assume active users or controlled environments, limiting their applicability in passive or real-world settings [31]. However, mobile devices generate traffic even when users are inactive, due to background activities such as updates, synchronization and telemetry. This raises the question of whether passive background traffic alone can reveal sufficient information for tracking and fingerprinting, as it is lower in volume and variability and therefore harder to exploit. Moreover, while modern MAC address randomization assigns different identifiers per network and defeats traditional identifier-based cross-location tracking, it does not fully prevent device identification, as devices can still be fingerprinted based on wireless metadata and protocol characteristics [36, 57].

In this paper, we present a traffic fingerprinting attack that requires no user interaction. Using multiple user profiles with diverse application sets and different devices, we collect traffic in a realistic Wi-Fi setting, and using only the extracted traffic metadata, we show that fingerprinting remains feasible without any user activity, and this enables user tracking across locations and over time, even if users randomize their MAC addresses [11, 23, 15].

The attack is particularly concerning in shared public networks such as malls or university campuses, where a passive adversary can link observations collected at different locations and times. The potential victims are ordinary users who might move between independent networks, such as from train station to a university. The attack exposes users' movement patterns, repeated visits, and behavioral routines without user interaction or awareness. The adversary can be a passive network eavesdropper that aims to track employee movements across the corporate network or organizations that track people participating in protests.

Existing passive fingerprinting approaches are either based on timing in which the device is identified based on inter-arrival times of frames or clock skew [40, 55], or classifiers of traffic to recognize the flow produced by applications [52]. However, both groups need to have the target enrolled beforehand to be able to decide in a close-world classification and the question we aim to answer has remained open whether two captures recorded at different times and places can be linked to the same person without any prior signature of that person. Moreover, in contrast to existing traffic analysis approaches that rely on machine learning (ML) and deep learning techniques [42, 46, 45, 38], our attack relies on a simple distance metric and communication endpoints, and it does not require computing resources and costly retraining due to concept drift (shortcomings of many ML-based attacks [27]). Specifically, we make the following contributions:

- We introduce passive background traffic fingerprinting, a traffic analysis attack that does not require user interaction and relies only on smartphone background traffic.
- We conduct a comprehensive experimental evaluation using more than 400 hours of traffic and identify factors influencing the attack. Our attack achieves $F1$ score of 0.89 and an accuracy of 0.99 in a closed-world setting.

– Unlike similar attacks that rely on complex machine-learning models, our attack uses simple distance-based metrics, is extremely lightweight, and does not require extensive traffic data.

2 Preliminaries

We study the feasibility of a traffic analysis attack that identifies similarities between user devices across observations, and uses passive background traffic recorded at multiple locations and times. We first state the problem and then provide the necessary technical background underlying the attack.

2.1 Problem Statement

We consider a set of WiFi networks at different physical locations, each serving a dynamic set of mobile users who connect for limited durations. Over time, a user might connect to multiple networks, forming a history of connections. Due to MAC address randomization, persistent link-layer identifiers might be unavailable, so the adversary cannot associate sessions or distinguish devices by their MAC address. Although a randomized MAC address may stay stable within a single network depending on the randomization mode, this stability is not guaranteed, as platforms such as iOS periodically rotate it [7], and a device uses a different (randomized) MAC address on each network, preventing cross-location tracking at the link layer. This limitation motivates the need for alternative techniques based on traffic analysis, which we present in this paper.

For a given network and observation interval, the *user set* is the set of all devices communicating during that observation period. We model each user device by a profile $u \in \mathcal{U}$, characterized by three significant attributes: device type, operating system, and installed applications. These attributes influence the background communication behavior of the device. Our attack focuses on *passive background* traffic, which is packets a device sends and receives autonomously without direct user interaction. We consider the traffic to be encrypted, so the adversary can only access metadata such as IP addresses of endpoints (§ 2.2).

2.2 Traffic Analysis Background

Traffic analysis attacks infer sensitive information from the observable characteristics of network transmissions, rather than their content [42, 41, 38]. We focus on passive fingerprinting [46, 42], where the adversary monitors traffic without interfering. Even under encryption, exploitable metadata such as packet headers and derived features remains available for traffic analysis, which the adversary assigns to individual profiles to eventually distinguish users. We describe the most relevant metadata features below. We note that availability of these features may differ on anonymous communication networks such as Tor [19].

IP Addresses. Source and destination IPs indicate connection endpoints. Factors such as load balancing, distributed infrastructures, or address changes over time may cause different IP addresses for the same hosts.

Autonomous Systems. Every endpoint belongs to an autonomous system (AS) governing inter-provider routing. An AS spans many IP addresses, enabling the adversary to map dynamic IP addresses to a relatively stable AS number.

Ports. Ports act as gateways that direct data correctly between devices. Client-side ports are ephemeral, but the server-side (well-known) ports indicate the service type (e.g., 443 for TLS) and are useful to the adversary (§5.3).

Additional Information. Packet sizes, timings, directions, and unencrypted headers have been used in prior (often ML-based) attacks [13, 24, 44, 12, 38].

2.3 Android Specifications

We focus on Android devices, whose operating system and vendors' implementations influence the traffic, both with and without user interaction. For example, Android's battery management system can limit background network access and periodically terminate background applications through a Least Recently Used (LRU) mechanism [5]. In a low power state like Doze mode, background execution is further restricted and applications cannot initiate arbitrary activities, and this is also affected by their startup state [3]. However, the system may wake processes and grant network access during periodic maintenance windows [4].

3 Attack

The adversary aims to link network traces of the same user device across different physical locations using passive background traffic generated without user interaction such as periodic synchronization. This restriction is intentional as it limits the identification signal to the ones contributed by the activities of the installed applications and filters out user behavior. The adversary collects traces at multiple locations and times, each capturing a device's network activity within a time window. In an offline phase, the adversary compares traces from different locations by applying a distance metric to pairs of trace representations and comparing the result against a threshold; a distance below the threshold indicates that traces originated from the same device. Matching related traces lets the adversary track devices across locations and infer behavioral patterns over time. We next describe the threat model and the attack phases.

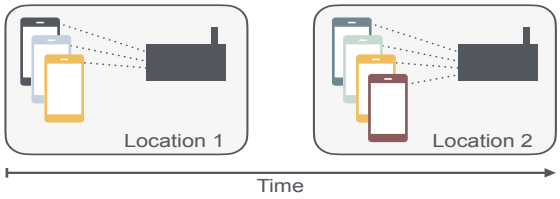

Fig. 1. The adversary records traffic traces in multiple locations (networks) and tries to link users' traces across locations.

3.1 Threat Model

Assumptions and Adversary Capabilities. The adversary passively monitors the wireless traffic of mobile devices in public environments such as cafés or other open access WiFi deployments, either by sniffing open, unencrypted networks or by deploying a malicious access point. Traffic collection occurs at multiple locations and times. The adversary captures network-layer metadata such as endpoint IP addresses, which is not protected by link-layer encryption (e.g., WPA2), as we assume an open network. We note that *not* all such metadata is persistent or identifying. MAC addresses, for instance, may be randomized or periodically rotated and thus cannot serve as persistent identifiers.

3.2 Attack Workflow

The attack has two main phases: data collection and trace linking.

Collection Phase. The adversary monitors a WiFi access point at two or more locations, passively recording all traffic from connected devices during a limited period (Figure 1). Each observed device is distinguished by a transient identifier derived from its MAC address during a single session. If MAC address rotation (due to randomization) happens mid-session, the adversary may attempt to relink the new MAC to the same device by checking the continuity in the 802.11 sequence numbers. When randomization is enabled, a different MAC address is used for each session and location (§2.1), so identification based on them across sessions and locations is not possible. We focus on a two location scenario for the clarity of notation: $\mathcal{T}^{(1)} = \{T_1^{(1)}, T_2^{(1)}, \ldots, T_{n_1}^{(1)}\}$ and $\mathcal{T}^{(2)} = \{T_1^{(2)}, T_2^{(2)}, \ldots, T_{n_2}^{(2)}\}$, where $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$ denote the traces collected at the first and second locations, respectively. The attack compares each pair of traces independently, so additional locations still reduce to additional pairwise comparisons and do not change the procedure.

Trace linking. The adversary analyzes the collected traffic to link traces of the same device. In preprocessing, captured traffic is transformed into a feature-based metadata representation (§5.1), where each trace is a set of timestamped metadata features from observed flows. For two traces T_i and T_j, the adversary computes a similarity score $sim(T_i, T_j)$ using a predefined distance over the features (§5.2), and a decision function determines whether they originate from the same device (§5.4). Applied to all pairs, this procedure results in matched pairs that link device sessions across locations and reveal user mobility patterns.

4 Experimental Setup

We build an experimental setup to implement and evaluate the attack, through producing the dataset of traffic traces for different user profiles.

4.1 Setup

The experimental setup consists of a mobile phone and a WiFi access point (Raspberry Pi 3 with `wpa_supplicant` [34]) that captures traffic using `tcpdump` [22]. The first round of experiments (January 2023) used a POCO F2 Pro (Android 10); the second (October 2024) used a Samsung Galaxy A14 5G (Android 14) to test the attack feasibility on a different vendor with a newer Android version.

User Profiles. To represent different users, we create each profile as a distinct set of installed applications, drawn from top 50 apps across 31 Google Play Store categories (sampled December 2022 and September 2023). Each profile's size is randomly chosen from [50, 100] (mean 80, std 20), and an app may appear in multiple profiles. Each category receives a popularity score from public statistics ([6]) (e.g., download counts, ratings), normalized to a probability distribution. Since the selected categories' scores sum to less than 1, a profile's number of apps falls below its target; we fill the remainder with the "communication" category, as mobile phone users often have a wide range of them installed.

Sampling Procedure. A profile is generated by (i) determining its size randomly from $[50, 100]$; (ii) selecting a weighted random subset of categories by popularity score; (iii) distributing the size across these categories (at least one app per category); (iv) picking random apps from each category according to numbers determined in step (iii); and finally (v) filling the remaining slots with random "communication" apps.

We generate 40 profiles from 473 applications, averaging 75 apps per profile (a few selected apps could not be installed, lowering the average number of apps). Profiles share between 4 and 33 applications, with an average overlap of 16.

App Activation and Traffic Recording. We factory reset the phone, disable as many pre-installed applications as possible, and install all apps needed across profiles. Applications are initially disabled and are activated per profile using `adb`. To capture a profile's background traffic, we launch its apps once to trigger initialization, ensuring each app completes its cold start (§2.3) so it can later sleep and resume while idle. We then close the apps, turn off the screen, and leave the device connected and idle while recording traffic for one hour. Afterward, the apps are disabled and the process repeats for the next profile.

4.2 Recording Repetitions.

To increase dataset diversity, we record multiple repetitions per profile, randomizing the app activation order each time to mimic realistic behavior of users (different order of using applications). We keep the default power management settings, as most users leave them unchanged. With one hour recording per profile (using `tcpdump`), 40 profiles, and 10 repetitions each, this yields 400 hours of traffic that serve as input for the attack and evaluation.

5 Attack Procedure

Below, we describe the four main steps of the attack: metadata extraction, endpoint processing, uniqueness analysis, and trace comparison.

5.1 Metadata Extraction

The recorded traffic metadata is the attack's main input. The adversary processes raw traces and extracts the header field information. Although features such as timing and protocol type are available, our attack focuses on connection endpoints, since they can reflect a user's installed apps and might be user-specific.

5.2 Endpoint Sets and Filtering

For each endpoint IP address in captured data, we resolve its autonomous system (AS) via Maxmind GeoIP database[1], representing each endpoint by its IP address and AS name. Then we remove endpoints with very low traffic volume, which tend to be outliers, using the standard Tukey formula for threshold $t_v = Q_1 - 1.5 * IQR$ [54].

5.3 Endpoint Uniqueness

An endpoint's distinguishing power depends on how rare it is; an endpoint appearing in all profiles has no distinguishing power. We assign each AS a commonness value, where a lower value indicates higher uniqueness and more significance for the decision. The commonness of an AS is the average of three characteristics: (i) the number of distinct profiles communicating with the AS; (ii) the destination ports these profiles use; and (iii) the AS's packet size diversity.

When fewer profiles communicate with an AS, the AS can be considered more unique. The fewer ports being used to communicate with an AS shows that the behavior of that AS is more specific. Finally, lower packet-size variability indicates more consistent, distinctive behavior. To measure packet-size diversity of an AS, we categorize its packet sizes into ten bins from 0.1 (least diverse) to 1.0 (most diverse), compute this per AS–profile pair, and average over profiles. Since equal averages can hide different variability, we take a weighted average over the range between the AS's lowest and highest bins. We prefer this to metrics such as standard deviation because, for behavioral consistency, the extreme bins matter more than their distribution. For example, "Fastly Inc." is used by 37 profiles, on average 37 for the same purpose (same ports), with average packet-size diversity 7.4, giving a commonness of 27.1; "Opera Software Americas LLC" on the other hand scores 1, 1, and 0.07 respectively, which gives it a commonness value of 0.6.

Uniqueness Decision. The commonness value decides whether an endpoint's IP or AS information is used in the attack step. To this end, we define a

[1] https://www.maxmind.com/en/geoip-databases

commonness threshold $t_c = Q_3 + 1.5 * IQR$ that detects outliers (similar to §5.2) and defines in which cases the AS cannot contribute to a clearer decision. The information of a sufficiently unique AS is kept; otherwise we extend it with port information into AS-port strings (excluding the ubiquitous HTTPS port 443). For example, "Amazon.com, Inc" is used by 37 profiles, all over HTTPS and port 5222, but only one profile over port 123; so the string "Amazon.com, Inc.123" (and other informative ports of the AS) is used instead of the AS name.

5.4 Trace Comparison

After preprocessing (§5.1 to §5.3), each trace is represented by its communication endpoints, and each endpoint can be an IP address, an AS, or the combined AS-port (§5.3). Comparing traces provides information on pairwise similarity that lets the adversary infer which traces are related and track a user across locations over time. This requires two steps: (i) distance calculation, where we compute the Jaccard distance over the presence or absence of endpoints; and (ii) similarity check, where a distance threshold determines whether two traces are related.

Each trace consists of categorical endpoint identifiers that are drawn from a large pool (universe), so the comparison is between two sparse binary vectors, and shared absence here carries no information. In other words, two profiles not contacting a specific AS have nothing meaningful in common. This filters out the metrics that count absences as agreement and are affected by the size of the pool rather than profile's behavior (such as Hamming or Euclidean). Jaccard distance here can measure what fraction of endpoints contacted in either trace were contacted by both. Over all candidate pairs, a distance lower than the threshold shows the traces are deemed to originate from the same user.

6 Evaluation

In this section, we analyze the performance of the attack using random repetitions and different attack parameters such as similarity threshold.

6.1 Evaluation Procedure

The evaluation procedure consists of two steps which are explained as follows.

Preparation. We obtain the raw dataset through traffic measurements and extend endpoints with AS and port information, producing an extended dataset of labeled (profile, repetition) traces of endpoint information (IP address, AS, AS-port). Mirroring a real attack, this dataset contains both the training information an adversary gathers beforehand, and the traces of unidentified victims.

Random Repetitions. For each repetition: (i) we split the whole 400 traces into training (80%) and testing (20%) sets, and we remove test profiles from the training set to avoid bias. This separation ensures that no trace from the same profile occurs both in train and test sets, which would give an unrealistic advantage to the attacker. (ii) We then choose a random starting point in time for

each trace, and extract a fixed-length time window with a different starting point from all traces. This is done to allow an observation window shorter than one hour, and to use different data for each repetition. (iii) Next, we compute commonness values and find the optimal decision threshold on the training subset. (iv) Finally, we perform pairwise trace comparison on the test subset. It should be noted that training and testing sets are not used in the machine-learning sense, and in our method, no machine learning model is fitted.

6.2 Evaluation Metrics

We classify the decisions as in Table 1 and compute the accuracy ($(TP + TN)/(TP + TN + FP + FN)$) for overall correctness, but the dataset is highly imbalanced since only a small fraction of trace pairs are related. Table 2 shows that on average only 7% of trace pairs match, while the remaining 93% are unrelated. Hence, we also use precision, recall, and F_1 score ($2 \cdot (precision \cdot recall)/(precision + recall)$) to evaluate performance.

Table 1. Decision Classes.

Result	**Label**	**Guess**	**Outcome**
$dist > t$	Match	Other	False Negative
$dist > t$	Other	Other	True Negative
$dist < t$	Match	Match	True Positive
$dist < t$	Other	Match	False Positive

Table 2. Average Ground Truth Matches.

	Mean	**Std**	**Min**	**Max**
Related Pairs	141	56	53	273
Unrelated Pairs	2360	360	1662	3027
Total	2506	377	1805	3148

6.3 Factors Influencing the Attack Success

We study how several factors affect the accuracy of the attack, then derive the final attack setup used in the main evaluation.

Recording Duration. The recording duration determines how much information is available. We test 500–3540 seconds with random offsets; longer recordings can expose more endpoints and improve performance (§6.4).

Decision Threshold. We use a threshold to judge whether two traces originate from one user; a misconfigured value inflates false positives or negatives. We divide the range of computed distances into 10 equal intervals, giving 10 candidate thresholds in [0, 1] (the Jaccard range), and select the one maximizing F_1 against the ground truth.

Endpoint Representations. We implement the attack with three endpoint sets: ASes only, IP addresses only, and the combination of ASes and AS-port strings (where an AS is non-unique).

Applications and Devices. To show that the results are not tied to specific profiles or hardware, we repeat the attack on a second phone (Samsung Galaxy A14 5G, Android 14) with new profiles from an updated app list (§4.1).

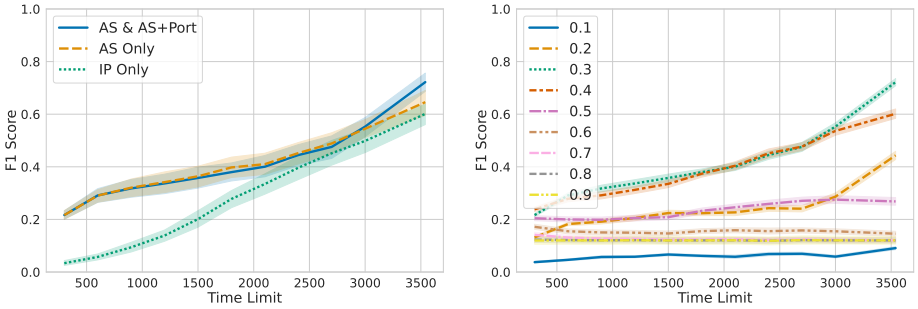

Fig. 2. Left: The plot shows attack performance (F_1 score) in different observation time limits (seconds) and allows us to compare the quality of different types of endpoints. The combination of AS and AS-port strings leads to the best performing attack setting. Right: The plot compares the average F_1 score in different observation time limits (seconds) considering different decision thresholds. It indicates that a threshold of 0.3 leads to the best attack performance in all tested time limits. Other thresholds are either too restrictive or relaxed, causing false positive or false negative decisions.

Table 3. Evaluation of the attack using 30 repetitions.

	5 Minutes				**30 Minutes**				**60 Minutes**			
	Mean	Std	Min	Max	Mean	Std	Min	Max	Mean	Std	Min	Max
Accuracy	0.93	0.01	0.90	0.96	0.93	0.01	0.91	0.95	0.97	0.01	0.94	0.99
Precision	0.30	0.10	0.14	0.46	0.38	0.15	0.16	0.68	0.85	0.09	0.67	0.98
Recall	0.18	0.05	0.08	0.30	0.35	0.10	0.14	0.57	0.63	0.14	0.30	0.89
$F1$ score	0.21	0.05	0.14	0.34	0.35	0.10	0.17	0.54	0.72	0.09	0.40	0.89

6.4 Experiments

In the first phase, we vary one factor per experiment while fixing the rest, run 30 random repetitions each, and retain the best performing setting. Our main observations regarding the influencing factors are as follows:

Recording time is a tradeoff between attack overhead and success, so rather than an optimal value we report the attack success over a range. Figure 2 (right) shows that across time limits, 0.3 is the optimal decision threshold as it gives the best $F1$ score, balancing false positives and negatives.

Table 3 summarizes performance over 30 simulation rounds on the POCO phone for different time limits at the optimal threshold of 0.3. The results show that although the constantly high accuracy (0.93-0.97) reflects data imbalance rather than detection ability, the attack success still improves sharply with the increase of observation time, with the average $F1$ score of 0.21 for 5-minute increases to 0.72 for one-hour traces, with the possibility of reaching to 0.89. Figure 2 shows that using IP addresses results in the lowest F_1 score, whereas AS information or AS-port strings achieve higher success.

Finally, Figure 3 (left) summarizes performance for each device alone and for the combination of two phones (cross-device matching); the attack performs

well in all setups. The relative values in the right plot are the normalized average of how many distinct ASes, IP addresses, and packets exists in different observation time limits for each phone over 30 repetitions of the attack. The two plots illustrate that the attack against the second phone (Samsung) with more up-to-date Android version (14 vs 10) achieves higher success due to the larger amount of network traffic sent to/from the phone, providing more signals to the adversary even without user interaction. The same reasoning applies to the higher success rate of the attack in the observations with longer durations, as it provides the attacker with more identifying signals.

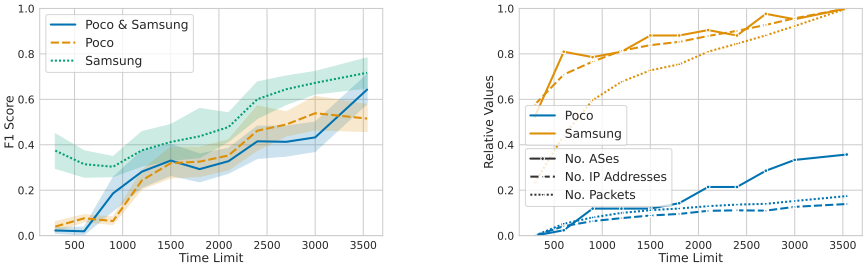

Fig. 3. Left: The comparison of attack performance ($F1$ score) for two devices and their combination within different observation time limits (seconds) shows that the attack feasibility does not rely on the device. Right: Relative values on the y-axis represent the normalized counts of distinct ASes, IP addresses, and packets observed across different time limits (seconds), scaled to the maximum observed value for each metric. Larger volume of endpoints and packets available to the attacker may explain the higher success rate for the Samsung phone (Android 14), and for longer observations.

7 Discussion

Below we discuss the limitations of our work and possible countermeasures.

7.1 Limitations

Our real-world assumptions and chosen setup introduce the following limitations.

Link-Layer Encryption. As the attack relies on metadata such as IP/AS endpoints, it cannot be run on link-layer encrypted networks (e.g., WPA2/WPA3). Nevertheless, a large fraction of public WiFi deployments remain unencrypted.

Training Phase. The adversary needs a dataset to compute endpoint commonness values, obtainable during the preliminary monitoring step. These values may become outdated over time [27], which may be addressed by periodic retraining.

Profiles. The limited number of profiles may understate difficulty in realistic settings with thousands of users, and profiles built from popular apps may not

match real ones. However, given millions of available apps, many users likely have a unique set of installed apps, which makes the attack feasible.

Tested Devices and OSes. Our setup uses only two Android smartphones and versions, but shared Android characteristics may generalize the attack to other Android devices. Future work can investigate whether iOS devices are vulnerable to similar attacks.

7.2 Countermeasures

Routing traffic via intermediate hops using VPNs and Tor can hide the endpoints observed at the access point. This may, however, increase latency and be vulnerable to more advanced traffic analysis attacks [25, 29]. A potential, but costly countermeasure could be to perturb profiles by sending cover traffic to additional endpoints , similar to defenses proposed against website fingerprinting attacks [16, 21].

8 Related Work

We review the state of the art in network traffic analysis, focusing on attacks on encrypted traffic and recent advances in device fingerprinting and tracking.

8.1 Traffic Analysis Attacks

Due to its security and privacy features, Tor is a prominent target of traffic analysis attacks [41, 45]. End-to-end confirmation attacks monitor the first and last hop of a circuit to match a client with a server [38, 39, 42]. Despite high reported success, the evaluation assumptions of end-to-end confirmation attacks may be unrealistic [46].

Website fingerprinting commonly train a ML model based on traffic features [43, 45, 48, 49, 12]. While website fingerprinting works even in the presence of link-layer encryption, many such attacks rely on unrealistic assumptions [27], such as their dependence on active user interactions, which must be modeled and simulated, and affects the validity of results. In contrast, our work uses passive background traffic and avoids reliance on active user interaction.

8.2 Deep Learning in Traffic Analysis

Deep learning has been applied to protocol detection [59], application identification from encrypted traffic [1], analysis of encrypted streaming data [47], and more practical settings such as app usage detection in cellular networks, where encrypted LTE/5G traffic still reveals application behavior [58]. An orthogonal line of research highlighted the limitations of these approaches, including generalization issues, dependency on large training datasets, and sensitivity to distribution shifts [8, 37, 14, 18]. In contrast, we show that effective fingerprinting is achievable with simple distance-based methods, without complex models.

8.3 Fingerprinting, Profiling, Tracking

Neumann et al. [40] compared histograms of 802.11 inter-arrival times of frames and Uluagac et al. [55] used clock skew on the wired backbone networks to classify devices. Both attacks relied on hardware or channel conditions, and they identified device models rather than users. Taylor et al. [52] identified the applications behind a flow from packet features, which required labelled training data for targets. Meneghello et al. [35] presented a passive traffic fingerprinting attack exploiting 5G/LTE control-channel information to identify smartphones and infer their location. Li et al. [33] proposed a graph learning framework for user profiling that, by modeling users, applications, locations, and time from spatiotemporal app usage data, inferred characteristics such as gender and age. Bahuguna et al. [10] introduced a deep-learning framework to identify mobile applications from encrypted traffic, and recognized user activities and user profiles. The WiFinger system used packet-level sequence matching to identify IoT events from noisy WiFi captures [31]. Todt et al. inferred user identities using channel characteristics broadcast by clients in WiFi 5 networks, namely beamforming feedback information [53]. Other researchers proposed methods to identify mobile devices despite modern protections such as MAC randomization, under realistic wireless network settings [36, 57]. Recent work also showed that devices and users remain distinguishable even under encryption and identifier obfuscation [26, 17].

While these works address device identification, event detection, or physical-layer leakage, our work links user traces across locations from passive background traffic. Our approach does not require active user interaction, explicit identifiers, or special hardware. Moreover, our attack does not rely on training compute-intensive ML models, which need to be periodically retrained due to cope with concept drift.

9 Conclusion

We introduced a passive background fingerprinting attack that uses background traffic generated automatically by mobile devices without requiring active behavior, which makes it more realistic compared to previous works. We show that even in the absence of payload access and persistent identifiers, metadata provide sufficient signals to distinguish user profiles and link traces across locations.

Our analysis identifies the main factors that influence performance, including endpoint representation, recording duration, and device characteristics. We show that autonomous system (AS) information, combined with transmission port, enables effective differentiation of user profiles, and longer observation periods improve the reliability of trace matching. Furthermore, the attack remains effective across different devices and application sets, highlighting its robustness. As a result, our findings show that passive background traffic alone can enable cross-location tracking of users, even when common privacy protections such as encryption and MAC randomization are in place. This highlights a gap in ex-

isting defenses and underlines the need for stronger protection mechanisms that address metadata leakage at the network level.

Data and Code Availability.

The code implementing the attack and analysis pipeline can be accessed at https://github.com/shivaz22/fingerprinting-2026. The captured traffic traces are not published by default, as raw packet captures may contain sensitive information. A sanitized version of the dataset is available on request from the corresponding author.

References

1. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A.: Mobile encrypted traffic classification using deep learning. In: 2018 Network traffic measurement and analysis conference (TMA). pp. 1–8. IEEE (2018)
2. Agrawal, A., Bhatia, A., Bahuguna, A., Tiwari, K., Haribabu, K., Vishwakarma, D., Kaushik, R.: A survey on analyzing encrypted network traffic of mobile devices. International Journal of Information Security **21**(4), 873–915 (2022)
3. Android Developers: App startup time (2024), https://developer.android.com/topic/performance/vitals/launch-time, accessed: 2026-04-10
4. Android Developers: Optimize for doze and app standby (2024), https://developer.android.com/training/monitoring-device-state/doze-standby, accessed: 2026-04-10
5. Android Developers: Processes and app lifecycle (2024), https://developer.android.com/guide/components/activities/process-lifecycle, accessed: 2026-04-20
6. AppBrain Statistics: Most popular google play categories (2022), https://www.appbrain.com/stats/android-market-app-categories, accessed: 2022-12-20
7. Apple: Use private Wi-Fi addresses on Apple devices - Apple Support (Dec 2025), https://support.apple.com/en-us/102509, online; accessed 20. Apr. 2026
8. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 3971–3988 (2022)
9. Auliya, S., Nugroho, L.E., Setiawan, N.A.: A review on smartphone usage data for user identification and user profiling. Communications in Science and Technology **6**(1), 25–34 (2021)
10. Bahuguna, A., Agrawal, A., Bhatia, A., Tiwari, K., Vishwakarma, D.: User profiling using smartphone network traffic analysis. In: 2021 International Conference on COMmunication Systems & NETworkS (COMSNETS). pp. 69–73. IEEE (2021)
11. Bar-Noy, A., Kessler, I.: Tracking mobile users in wireless communications networks. IEEE Transactions on Information Theory **39**(6), 1877–1886 (1993)
12. Bhat, S., Lu, D., Kwon, A., Devadas, S.: Var-cnn: A data-efficient website fingerprinting attack based on deep learning. arXiv preprint arXiv:1802.10215 (2018)

13. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy vulnerabilities in encrypted http streams. In: International Workshop on Privacy Enhancing Technologies. pp. 1–11. Springer (2005)
14. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)
15. Clarke, R., Wigan, M.: You are where you've been: The privacy implications of location and tracking technologies. Journal of Location Based Services **5**(3-4), 138–155 (2011)
16. Cui, W., Yu, J., Gong, Y., Chan-Tin, E.: Realistic cover traffic to mitigate website fingerprinting attacks. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). pp. 1579–1584. IEEE (2018)
17. Cunha, M., Mendes, R., de Montjoye, Y.A., Vilela, J.P.: On the difficulty of not being unique: Fingerprinting users from wi-fi data in mobile devices. In: Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing. pp. 1335–1344 (2025)
18. Dacrema, M.F., Cremonesi, P., Jannach, D.: Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: Proceedings of the 13th ACM Conference on Recommender Systems. pp. 101–109 (2019)
19. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second generation onion router. In: Usenix Security. vol. 10. USENIX Association Berkeley, CA (2004)
20. Du, J., Xu, H., Tu, Z.: Personalized recommendation model of traffic package based on user consumption behavior. In: 2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP). pp. 1–6. IEEE (2019)
21. Gong, J., Wang, T.: Zero-delay lightweight defenses against website fingerprinting. In: 29th USENIX security symposium (USENIX security 20). pp. 717–734 (2020)
22. Group, T.T.: tcpdump and libpcap. https://www.tcpdump.org/ (2024)
23. Gruteser, M., Liu, X.: Protecting privacy, in continuous location-tracking applications. IEEE Security & Privacy **2**(2), 28–34 (2004)
24. Hayes, J., Danezis, G.: k-fingerprinting: A robust scalable website fingerprinting technique. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 1187–1203 (2016)
25. Heijligenberg, T., Lkhaouni, O., Kohls, K.: Leaky blinders: Information leakage in mobile vpns. In: International Conference on Applied Cryptography and Network Security. pp. 481–494. Springer (2022)
26. Huang, Y., Dong, Z., Yang, X., Zhang, D., Wang, Q., Wang, Z.: Smartphone user fingerprinting on wireless traffic. IEEE Transactions on Mobile Computing (2025)
27. Juarez, M., Afroz, S., Acar, G., Diaz, C., Greenstadt, R.: A critical evaluation of website fingerprinting attacks. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. pp. 263–274 (2014)
28. Kambar, M.E.Z.N., Esmaeilzadeh, A., Kim, Y., Taghva, K.: A survey on mobile malware detection methods using machine learning. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference. pp. 0215–0221
29. Kohls, K., Rupprecht, D., Holz, T., Pöpper, C.: 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. pp. 249–260 (2019)
30. Li, J., Zhou, H., Wu, S., Luo, X., Wang, T., Zhan, X., Ma, X.: FOAP:Fine-Grained Open-World Android App Fingerprinting. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 1579–1596 (2022)
31. Li, R., Liu, S., Hu, H., Ye, Q., Feamster, N.: Wifinger: Fingerprinting noisy iot event traffic using packet-level sequence matching (2025)

32. Li, T., Li, Y., Hoque, M.A., Xia, T., Tarkoma, S., Hui, P.: To what extent we repeat ourselves? discovering daily activity patterns across mobile app usage. IEEE Transactions on Mobile Computing **21**(4), 1492–1507 (2020)
33. Li, T., Li, Y., Zhang, M., Tarkoma, S., Hui, P.: You are how you use apps: user profiling based on spatiotemporal app usage behavior. ACM Transactions on Intelligent Systems and Technology **14**(4), 1–21 (2023)
34. Malinen, J.: Linux WPA/WPA2/WPA3/IEEE 802.1X Supplicant (2013), https://w1.fi/wpa_supplicant/
35. Meneghello, F., Rossi, M., Bui, N.: Smartphone identification via passive traffic fingerprinting: A sequence-to-sequence learning approach. IEEE Network **34**(2), 112–120 (2020)
36. Mishra, A.K., Pélissier, S., Cunche, M.: Fingerprinting connected wi-fi devices using per-network mac addresses. In: International Symposium on Foundations and Practice of Security. pp. 295–303. Springer (2024)
37. Musgrave, K., Belongie, S., Lim, S.N.: A metric learning reality check. In: European Conference on Computer Vision. pp. 681–699. Springer (2020)
38. Nasr, M., Bahramali, A., Houmansadr, A.: Deepcorr: Strong flow correlation attacks on tor using deep learning. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. pp. 1962–1976 (2018)
39. Nasr, M., Houmansadr, A., Mazumdar, A.: Compressive traffic analysis: A new paradigm for scalable traffic analysis. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 2053–2069 (2017)
40. Neumann, C., Heen, O., Onno, S.: An empirical study of passive 802.11 device fingerprinting. In: 2012 32nd International Conference on Distributed Computing Systems Workshops. pp. 593–602. IEEE (2012)
41. Oh, S.E., Sunkam, S., Hopper, N.: p1-fp: Extraction, classification, and prediction of website fingerprints with deep learning. Proceedings on Privacy Enhancing Technologies **2019**(3), 191–209 (2019)
42. Oh, S.E., Yang, T., Mathews, N., Holland, J.K., Rahman, M.S., Hopper, N., Wright, M.: Deepcoffea: Improved flow correlation attacks on tor via metric learning and amplification. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1915–1932. IEEE (2022)
43. Panchenko, A., Lanze, F., Pennekamp, J., Engel, T., Zinnen, A., Henze, M., Wehrle, K.: Website fingerprinting at internet scale. In: NDSS. vol. 1, p. 23477 (2016)
44. Rahman, M.S., Sirinam, P., Mathews, N., Gangadhara, K.G., Wright, M.: Tiktok: The utility of packet timing in website fingerprinting attacks. arXiv preprint arXiv:1902.06421 (2019)
45. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning (2017)
46. Rimmer, V., Schnitzler, T., Van Goethem, T., Romero, A.R., Joosen, W., Kohls, K.: Trace oddity: Methodologies for data-driven traffic analysis on tor. Proceedings on Privacy Enhancing Technologies (2022)
47. Schuster, R., Shmatikov, V., Tromer, E.: Beauty and the burst: Remote identification of encrypted video streams. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1357–1374 (2017)
48. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. pp. 1928–1943 (2018)

49. Sirinam, P., Mathews, N., Rahman, M.S., Wright, M.: Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1131–1148 (2019)
50. Sung, K., Biswas, J., Learned-Miller, E., Levine, B.N., Liberatore, M.: Server-side traffic analysis reveals mobile location information over the internet. IEEE Transactions on Mobile Computing **18**(6), 1407–1418 (2018)
51. Sánchez, P.M.S., Valero, J.M.J., Celdrán, A.H., Bovet, G., Pérez, M.G., Pérez, G.M.: A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets. IEEE Communications Surveys & Tutorials **23**(2), 1048–1077 (2021)
52. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 439–454
53. Todt, J., Morsbach, F., Strufe, T.: Bfid: Identity inference attacks utilizing beamforming feedback information. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security. pp. 2399–2413 (2025)
54. Tukey, J.W., et al.: Exploratory data analysis, vol. 2. Addison-wesley Reading, MA (1977)
55. Uluagac, A.S., Radhakrishnan, S.V., Corbett, C., Baca, A., Beyah, R.: A passive technique for fingerprinting wireless devices with wired-side observations. In: 2013 IEEE Conference on Communications and Network Security (CNS). pp. 305–313
56. Velan, P., Čermák, M., Čeleda, P., Drašar, M.: A survey of methods for encrypted traffic classification and analysis. International Journal of Network Management **25**(5), 355–374 (2015)
57. Vogel, D., Viola, F., Kreimeyer, N.M., Meier, M.: I know you were here: Leveraging probe request templates for identifying wi-fi devices. In: 2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). pp. 429–436. IEEE (2024)
58. Wang, J., Cheng, Z., Ordean, M., Cui, B.: What-app? app usage detection using encrypted lte/5g traffic. Proceedings on Privacy Enhancing Technologies (2026)
59. Wang, Z.: The applications of deep learning on traffic identification. BlackHat USA **24**(11), 1–10 (2015)
60. Wang, Z., Wei, G., Zhan, Y., Sun, Y.: Big data in telecommunication operators: data, platform and practices. Journal of Communications and Information Networks **2**(3), 78–91 (2017)
61. Wu, T., Xiao, X., Li, Q., Liu, Q., Hu, G., Luo, X., Jiang, Y.: Behavsniffer: Sniff user behaviors from the encrypted traffic by traffic burst graphs. In: 2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). pp. 456–464. IEEE (2023)
62. Xu, Q., Zheng, R., Saad, W., Han, Z.: Device fingerprinting in wireless networks: Challenges and opportunities. IEEE Communications Surveys & Tutorials **18**(1), 94–104 (2015)