

# GPSniffer: Group-based Location Inference Attack in Augmented Reality using Network Traffic Analysis

Shuaikang Hou  
shou2@student.gsu.edu  
Georgia State University  
Atlanta, Georgia, USA

Srinivasan Murali  
smurali@tamsu.edu  
Texas A&M University-San Antonio  
San Antonio, Texas, USA

Mingjing Liang  
mingjing.liang@ucdconnect.ie

Mingyan Xiao  
mxiao@cpp.edu  
Cal Poly Pomona  
Pomona, California, USA

Huadi Zhu  
hzhu17@gsu.edu  
Georgia State University  
Atlanta, Georgia, USA

## Abstract

Augmented Reality (AR) applications increasingly rely on cloud-based processing, where camera frames are frequently uploaded for visual understanding and content generation. While such designs enable rich functionality, they also introduce new privacy risks through unintended side channels. In this paper, we present GPSniffer, a novel group-based location inference attack that exploits encrypted network traffic patterns generated by AR applications. The key insight is that co-located users capture similar visual scenes, resulting in correlated network traffic patterns that can be used to infer their physical proximity. GPSniffer leverages network metadata and zero-permission IMU sensors to identify co-located users. By propagating location information from a small subset of users who reveal their GPS data (*anchor users*), GPSniffer can infer the locations of a larger population of users (*target users*).

We implement GPSniffer and evaluate it using a dataset collected across multiple real-world locations. Experimental results show that GPSniffer achieves high inference accuracy (0.949), outperforming prior AR-specific attacks, while eliminating their reliance on pre-deployed artifacts. Our findings highlight a previously overlooked privacy vulnerability in immersive systems and call for new defenses against traffic-based inference attacks.

## CCS Concepts

• **Security and privacy** → **Mobile and wireless security**; **Mobile platform security**.

## Keywords

Extended reality, network analysis attack, location inference attack

### ACM Reference Format:

Shuaikang Hou, Srinivasan Murali, Mingjing Liang, Mingyan Xiao, and Huadi Zhu. 2026. GPSniffer: Group-based Location Inference Attack in Augmented Reality using Network Traffic Analysis. In *Second Workshop on Enhancing Security, Privacy, and Trust in Extended Reality (XR Security '26)*, October 26–30, 2026, Austin, TX, USA. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3842203.3844582>



This work is licensed under a Creative Commons Attribution 4.0 International License. *XR Security '26, Austin, TX, USA*

© 2026 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-2951-5/2026/10  
<https://doi.org/10.1145/3842203.3844582>

## 1 Introduction

Augmented Reality (AR) applications are rapidly gaining adoption across domains such as navigation, shopping, education, and entertainment. Modern AR systems increasingly rely on cloud-based processing, where captured camera frames are frequently uploaded to remote servers for tasks such as object recognition, scene understanding, and content augmentation. While this design enables powerful functionality on these mobile devices, it also introduces new privacy risks, particularly through unintended side channels.

Prior work has demonstrated that user locations can be inferred on general mobile devices without explicit GPS access by exploiting side channels such as sensor data, power consumption, and network traffic [2, 6, 8, 10, 12, 13, 15, 19]. These studies highlight the fundamental feasibility of tracking users without their consent, even under restrictive permission models. More recently, research has explored such attacks in AR settings [11, 17]. However, these state-of-the-art approaches rely on deploying specially crafted virtual artifacts at target locations before launching the attack, which limits their scalability and practicality in real-world scenarios. In parallel, a large body of work has shown that encrypted network traffic still exposes rich side-channel information, such as packet size, timing, and transmission patterns, which can be used to infer user activities or system states [3, 7, 16, 20]. Meanwhile, group-based inference techniques leverage correlations across users, where information from a subset of users can be used to infer properties of others [4, 5, 14, 18]. These insights suggest a new opportunity: combining traffic analysis with cross-user correlation to enable scalable location inference without requiring environmental manipulation.

In this paper, we present *GPSniffer*, a novel group-based location inference attack targeting AR applications. The key insight is that co-located users interacting with AR apps tend to upload visually similar frames of their surroundings, leading to correlated network traffic patterns—even when the traffic is encrypted. By identifying such correlations, GPSniffer can group co-located users and propagate location information from a small set of users who reveal their GPS data (referred to as *anchor users*) to a larger population who do not (*target users*). Figure 1 illustrates this process: Assume the attacker has already obtained GPS readings from anchor users (A, B, and C); by analyzing network traffic patterns, the attacker identifies that target user T is co-located with anchor user B, and thus infers T's location as B's GPS reading.

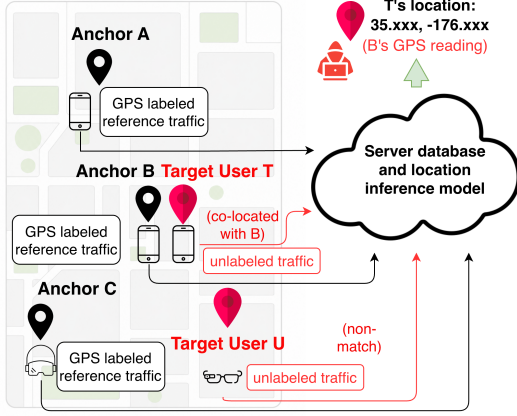


Figure 1: Attack illustration.

GPSniffer operates without requiring any prior environment setup, camera access, or semantic understanding of the transmitted data. It extracts robust features from encrypted network traffic via signal preprocessing and envelope extraction, filters candidate pairs using IMU- and time-based distance metrics, and performs co-location inference through a Siamese neural network. By relying solely on network metadata and the zero-permission IMU sensor, it remains stealthy and broadly applicable.

We evaluate GPSniffer through a real-world prototype and dataset collected across 10 different locations across 5 device models. Our results show that GPSniffer achieves better inference accuracy than prior AR-specific attacks while removing their reliance on pre-deployed artifacts.

In summary, this paper makes the following contributions:

- We identify a new side channel in AR systems by linking environmental visual content to network traffic patterns.
- We design GPSniffer, a group-based location inference framework leveraging cross-user correlations without prior setup.
- We implement and evaluate the system in real-world settings, demonstrating its effectiveness and generalizability.

Our findings reveal a previously overlooked privacy risk in AR and call for new defenses against traffic-based inference attacks.

## 2 Related Work

**Location inference attacks.** Prior work has demonstrated that user locations can be inferred on general mobile devices without explicit GPS access by exploiting side channels such as sensor data, GPU leakage, browser cache, or mobility regularities [2, 6, 8–10, 12, 13, 15, 19]. These studies highlight the fundamental feasibility of tracking users without their consent, even under restrictive permission models. Recent research has explored location inference in the context of augmented reality (AR) apps. Meyer-Lee *et al.* [11] shows that AR apps may leak location information through network traffic. Shang *et al.* [17] further demonstrates that multi-player AR apps can be exploited for user location tracking. However, these AR-specific attacks rely on deploying specially crafted virtual artifacts at targeted locations prior to launching the attack, which limits their generalizability and scalability in real-world settings. In contrast,

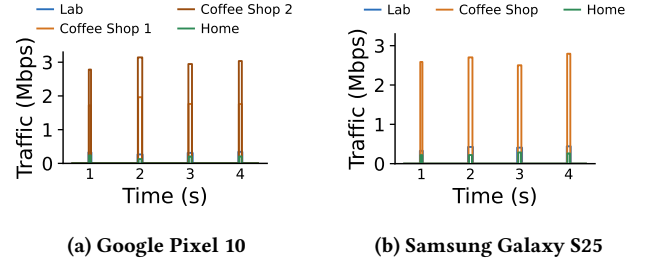


Figure 2: Feasibility study of network traffic patterns of an AR app at different locations across two devices.

this work studies a novel location inference attack using intrinsic network traffic patterns generated during AR interactions, without requiring any prior setup.

**Network analysis attacks.** A large body of work leverages encrypted network traffic for inference attacks, exploiting metadata such as packet size, timing, and direction. Website fingerprinting attacks (e.g., [7, 20]) demonstrate that user activities can be identified despite encryption. Meanwhile, app-level inference attacks (e.g., [3, 16]) reveal user actions, visited content, or device states from traffic patterns. These techniques demonstrate that even encrypted traffic retains rich side-channel information, which can be exploited to launch attacks such as location inference.

**Group-based attacks.** Group-based or correlation-based attacks leverage similarities across multiple users to infer sensitive information. For instance, [18] and [4] show that mobility traces can be linked across users based on spatiotemporal patterns. More recent work explores collaborative inference, where a subset of users with richer data (e.g., labeled or permission-granted users) is used to infer properties of others [5, 14]. These approaches demonstrate the power of cross-user correlation, which lays the foundation of the proposed attack framework.

## 3 Feasibility Study

The core idea of this work is to infer user location by exploiting network traffic patterns, even when encrypted, when using AR apps, which frequently upload camera frames to remote servers for processing. This work is based on the key assumption that the surrounding environment can influence the captured content, thus affecting the observed traffic patterns.

To validate this assumption, we conduct a small-scale feasibility study using the Google Lens app, which performs cloud-based visual analysis by uploading captured frames. We collect representative traffic traces from three locations: lab, coffee shop, and home, using two smartphone models: Google Pixel 10 and Samsung Galaxy S25. We fix the baseline user activities and the frame upload intervals to be identical across all sessions for fair comparison.

As shown in Figure 2, we make the following observations. First, different locations exhibit distinct network traffic patterns. For example, the coffee shop scene produces the highest peak rates during frame uploads, likely due to the complex scene structure and diverse visual content. Second, traffic patterns remain consistent at the same location (the same coffee shop), due to the similarity of the surrounding environment. Third, these patterns are consistent

across device models, suggesting generalizability across users with different devices.

Our work is inspired by these observations. When co-located users interact with AR applications, their devices tend to upload visually similar frames of the surrounding environment, leading to correlated network traffic patterns. By analyzing these patterns, we can identify groups of co-located users; consequently, if the GPS data of any of these users is compromised, it can be used to infer the locations of other users.

#### 4 Threat Model

**Attacker’s goal.** The attacker seeks to infer users’ locations while they are using AR apps without relying on their own GPS permission grants. Since AR devices are tightly coupled with the users’ physical positions, such inference effectively enables continuous user tracking.

**Attack scenarios.** The attacker distributes a malicious app via app marketplaces or repackaged software. The app is disguised as a benign service (e.g., food delivery or ride-sharing) and requests access to network usage data and, optionally, GPS data. Once the required permissions are granted, the app runs in the background and continuously collects and transmits these data to the attacker. Note that while GPS permission may be requested, the attack does not depend on it. Instead, it assumes an opportunistic setting where only a small fraction of users, referred to as *anchor users*, grant GPS permissions. These users, usually less vigilant ones, unintentionally become “spy nodes” and provide ground-truth location data that the attacker leverages to locate other users, referred to as *target users*, who do not grant GPS permissions.

**Attacker’s capabilities.** The attacker can collect network usage data from the device through a background application, a permission that is commonly granted by users and does not raise suspicion. The app may opportunistically obtain GPS data from a small subset of users, but such access is not required for the majority of victims. It may also use the zero-permission IMU sensors to refine location inference. However, the attacker cannot access camera data or any other restricted sensors. The attacker has no prior knowledge of the victims’ identities, locations, or mobility patterns. The attacker cannot decrypt application-layer traffic or extract semantic content from communications. Additionally, unlike many prior works [11, 17], the attacker does not rely on any pre-collection or site-specific calibration in target areas. Instead, the attacker passively uses network metadata and limited opportunistic GPS data alone.

#### 5 Attack Framework

In this paper, we introduce GPSniffer, an attack framework to infer locations of AR users without obtaining their GPS permissions. As shown in Figure 3, GPSniffer consists of three main modules. First, the attacker collects GPS data from anchor users, and network traffic signals from both anchor users and target users via the malicious app. Then, the collected network traffic signals are preprocessed and features are extracted from the clean signals. Finally, attacker adopts a matching-based location inference module to determine specific locations of target users. The rest of this section will elaborate on the technical design of each module.

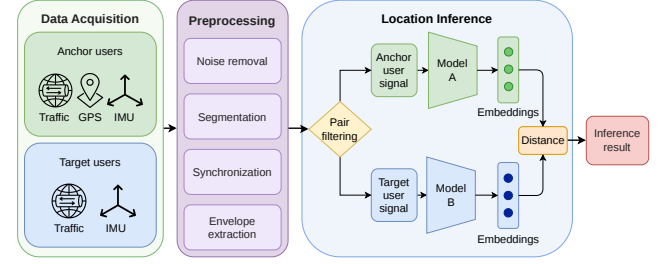


Figure 3: System architecture.

##### 5.1 Data Acquisition and Preprocessing

**Data acquisition.** Once users grant GPS permission to the malicious app, they are unintentionally turned into anchor users; otherwise, they are treated as target users. The attacker leverages GPS data from the small set of anchor users to label a larger population of target users by matching their network traffic patterns. To this end, GPSniffer collects data from both anchor and target users in real time as they interact with an AR application, with the malicious app running in the background. Specifically, it acquires GPS data from anchor users and network traffic signals, i.e., real-time app-specific upload rates, from both anchor and target users and uploads these data to the attacker’s server for analysis. Notably, many users do not carefully review app permissions or privacy notices prior to installation [1], making it feasible for the malicious app to collect such data without attracting user attention.

**Preprocessing.** Then, the network traffic signals undergo a preprocessing stage. First, we remove signal portions with abrupt fluctuations by applying a duration threshold  $\theta_d$  and a variance threshold  $\theta_v$  over a rolling window to identify sudden changes; these changes are typically caused by app switching or interface interactions and produce traffic unrelated to normal XR usage. Meanwhile, short bursts corresponding to data uploads are preserved. This process is formulated as:

$$s(t) = \begin{cases} 0, & \text{if } dur(s(t)) > \theta_d \text{ and } var(s(t)) > \theta_v \\ s(t), & \text{otherwise} \end{cases} \quad (1)$$

Here,  $dur(s(t))$  and  $var(s(t))$  denote the duration of the fluctuation and the local variance within the signal portion  $s(t)$ , respectively.

The remaining signal is then segmented using a sliding window with length  $w$  and step size  $\delta$ . Each segment is timestamped by its start time and, for anchor users, annotated with the corresponding GPS label to enable synchronization prior to matching. Each segment is treated as a basic analysis unit.

However, in real-world settings, the number and timing of frame uploads within each window are irregular, leading to unstable and misaligned features. To address this, we extract the signal envelope of each segment using the Hilbert Transform:

$$e(t) = \sqrt{x^2(t) + \hat{x}^2(t)} \quad (2)$$

with  $\hat{x}(t)$  denoting the Hilbert transform of the signal  $x(t)$ :

$$\hat{x}(t) = \mathcal{H}\{x(t)\} = \frac{1}{\pi} \text{P.V.} \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau \quad (3)$$

where P.V. represents the Cauchy principal value of the integral. The extracted envelope of each segment is then fed into the model for location matching.

## 5.2 Location Inference

The attacker selects a pair of segments, one from the anchor users and one from the target users, for location matching. If the embedding distance is below a predefined threshold, the target user is predicted to be co-located with the anchor user. Next, we describe the matching pair selection rule and the location matching approach.

**Matching pair selection.** Directly matching signals for location inference faces two practical challenges. First, variations in device orientation and motion can alter the captured scene, resulting in different network transmission patterns during XR interactions. Second, signals collected over extended time periods may differ due to environmental changes.

To address these issues, we define a distance score between any two signals to filter candidate pairs. We first leverage IMU sensors on the AR device to characterize device orientation and motion. Notably, IMU sensors are zero-permission sensors and can be accessed by any application without additional user approval. We compute the IMU distance as the weighted average of Dynamic Time Warping (DTW) distances across all channels:

$$D_i = \frac{1}{|C|} \sum_{c \in C} k_c \cdot \text{DTW}(\sigma_1^c, \sigma_2^c) \quad (4)$$

where  $C$  is the set of IMU channels,  $\sigma_1^c$  and  $\sigma_2^c$  indicate time series from channel  $c$  for the two segments being compared, and  $k_c$  is the weight for channel  $c$ . We also define the temporal distance  $D_t$  as the absolute difference between the starting timestamps of the two signals. The overall distance score is then given by  $D = D_i + \alpha \cdot D_t$ , where  $\alpha$  is a weighting coefficient. Signal pairs with a distance score exceeding a predefined threshold are discarded from matching.

**Siamese-based location inference.** To predict the similarity between two inputs, we employ a Siamese network consisting of two identical models with shared weights. We empirically adopt two 1D CNN models. Each model takes one signal as input and produces a latent embedding. The Euclidean distance between the two embeddings is then computed: if this distance is below a threshold  $\theta$ , the target user is inferred to be co-located with the anchor user. In cases where a target user matches multiple anchor users within a similar time window, the location of the anchor user with the minimum embedding distance is assigned to the target user.

**Model training.** The Siamese network is trained in a supervised manner using pairs of signal segments constructed from anchor users from our collected dataset. Positive pairs are formed from segments collected at the same location and within a close time window, and negative pairs are formed from segments collected at different locations. To learn discriminative embeddings, we adopt a contrastive loss that pulls positive pairs closer and pushes negative pairs apart in the embedding space:

$$\mathcal{L} = y \cdot \Delta(z_1, z_2) + (1 - y) \cdot \max(0, m - \Delta(z_1, z_2)) \quad (5)$$

where  $y \in \{0, 1\}$  indicates whether the pair is positive or negative,  $\Delta(\cdot, \cdot)$  is any distance function (we empirically adopt Euclidean distance),  $z_1$  and  $z_2$  are the embeddings, and  $m$  is a margin that

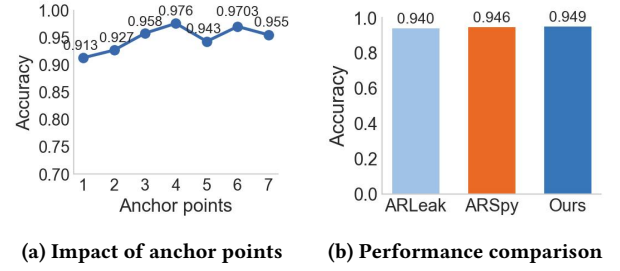


Figure 4: Overall performance of GPSniffer.

enforces separation between negative pairs. This objective encourages embeddings of co-located signals to have high similarity while ensuring that non-co-located signals remain sufficiently dissimilar.

## 6 Evaluation

### 6.1 Experimental Setup

To evaluate GPSniffer, we collect a dataset comprising 40 sessions across 10 locations in United States and China, using 5 different AR devices. We consider three representative foreground apps: Google Lens, which performs cloud-based visual analysis of the captured scene; Pokémon Go, a location-based AR game with distinct interaction patterns and traffic characteristics; and DescribeImage.ai, a image description service that uploads images for cloud-based visual understanding. We leave the exploration of additional AR apps to future work.

During each session, the malicious app runs in the background and records timestamps, encrypted network traffic, IMU readings, and GPS data (used as ground-truth labels) while the user interacts with the foreground app. We randomly split the dataset into a training set (80%) and a test set (20%). All subsequent evaluations are conducted on this dataset.

### 6.2 Overall Performance

We start by evaluating the overall performance of GPSniffer. As shown in Figure 4a, GPSniffer achieves high inference accuracy, consistently exceeding 0.913. The highest accuracy is achieved at 0.976 with 3 anchor users; the accuracy stabilizes around 0.96 as the number of anchor point grows. The average is 0.949. Note that our experiments are conducted at a limited scale (following the scale of prior work [11, 17]); in real-world deployments, the number of anchor users is expected to be larger. We leave large-scale evaluation to future work. We further compare GPSniffer against two state-of-the-art baseline approaches [11, 17]. GPSniffer achieves an overall accuracy of 0.949, outperforming both baselines (0.940 and 0.946, respectively). More importantly, GPSniffer does not require any prior environmental setup, which is a key assumption in both baseline approaches.

### 6.3 Consistency Study

To examine the generalizability of GPSniffer, we evaluate its performance across different settings below.

**Environment scenarios.** Figure 5a exhibits GPSniffer's performance across 6 representative environments: *indoor* – home,

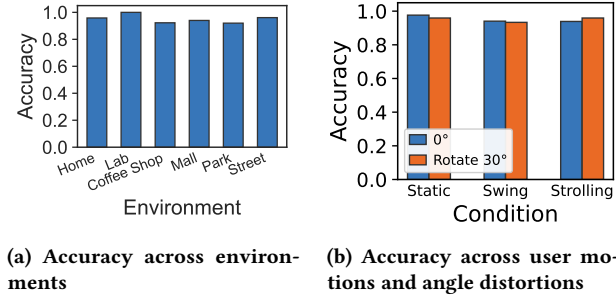


Figure 5: Consistency study of GPSniffer.

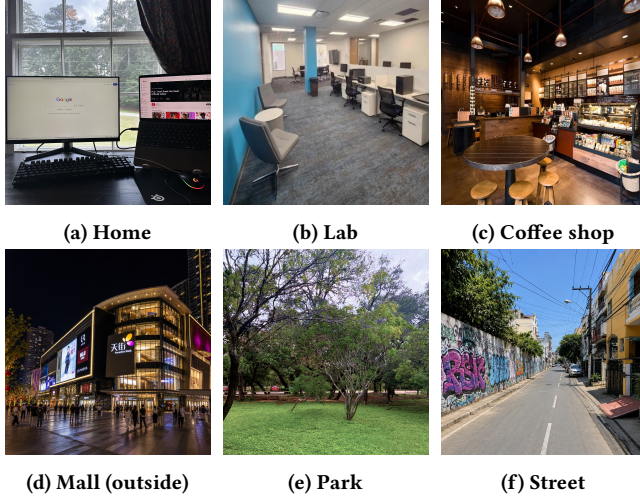


Figure 6: Representative environments.

Table 1: Performance across different devices.

| Device          | Type       | Acc.  | Prec. | Rec.  | F1    |
|-----------------|------------|-------|-------|-------|-------|
| Google Pixel 10 | Smartphone | 0.952 | 0.958 | 0.952 | 0.952 |
| Samsung S25     | Smartphone | 0.976 | 0.978 | 0.984 | 0.980 |
| Samsung A16     | Smartphone | 0.960 | 0.963 | 0.958 | 0.958 |
| Huawei P50 Pro  | Smartphone | 0.944 | 0.963 | 0.956 | 0.957 |
| Meta Quest Pro  | HMD        | 0.903 | 0.923 | 0.903 | 0.904 |

lab, coffee shop, *outdoor* – mall, park, and street. Figure 6 shows these environments. GPSniffer achieves consistently high accuracy across all environments, reaching 0.958, 1.000, 0.923, 0.940, 0.920, and 0.961 for home, lab, coffee shop, mall, park, and street, respectively. The accuracy is slightly lower in coffee shop and park, potentially due to more dynamic surroundings and less controlled user interactions. Overall, GPSniffer is generalizable across multiple daily environments.

**Device models.** We test five different AR devices, including four smartphones and a head-mount display (HMD), as shown in Table 1, at the above locations. We observe similar performance across all devices. Among smartphones, the accuracy ranges from 0.944 on Huawei P50 Pro to 0.976 on Samsung S25. On Meta Quest Pro,

Table 2: Performance across different applications.

| Application      | Type          | Acc.  | Prec. | Rec.  | F1    |
|------------------|---------------|-------|-------|-------|-------|
| Google Lens      | Visual search | 0.976 | 0.985 | 0.976 | 0.980 |
| Pokémon Go       | Mobile game   | 0.923 | 0.921 | 0.904 | 0.912 |
| DescribeImage.ai | Image search  | 0.903 | 0.932 | 0.903 | 0.904 |

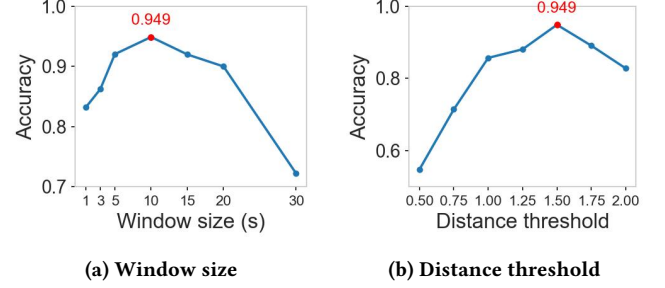


Figure 7: Impact of window size and distance threshold.

GPSniffer achieves an accuracy of 0.903. This indicates that the location-discriminative traffic patterns captured by GPSniffer are not specific to a single device.

**Applications.** We further evaluate GPSniffer across three different foreground AR applications: Google Lens, Pokémon Go, and DescribeImage.ai. Google Lens is a visual search application, Pokémon Go is a location-based AR game with different interaction patterns and traffic characteristics, and DescribeImage.ai is a web-based image description service where users upload images and receive visual descriptions. These representative applications are common for AR use. As shown in Table 2, GPSniffer achieves steady performance with an accuracy continuously above 0.903 and F1 over 0.904. This result suggests that GPSniffer can generalize across different types of AR application interaction.

**User motions.** We further evaluate GPSniffer under different user motion conditions since dynamic motion could impact the captured scene. We compare three common motion settings: stationary, swinging, and strolling. Results in Figure 5b suggests only minimal impact of motion on the attack performance, with the accuracy over 0.939 even in the worst condition. Therefore, performing user motions does not effectively address vulnerability against GPSniffer.

**Angle distortions.** Finally, we evaluate the impact of angle distortions, which could alter the captured scene content. We retest the above scenarios with a 30° rotation. As Figure 5b demonstrates, the accuracy is generally unaffected. This suggests that GPSniffer can tolerate moderate orientation changes while preserving major traffic signatures for location inference attack.

## 6.4 Micro Benchmark

**Window size  $w$ .** Window size  $w$  in §5 defines the length of the network traffic signal to be matched. We study its impact on location inference accuracy. As the window size increases from 1 s to 10 s, the accuracy improves from 0.832 to 0.949; further enlarging the window size leading to reduced accuracy (0.722 at 30 s). This is

because while small windows do not contain enough traffic context, overly large windows may smooth out the burst structure that is useful for distinguishing locations. Therefore, we choose the window size as 10 s.

**Distance threshold  $\theta$ .** We also evaluate the impact of the distance threshold  $\theta$  used in Siamese-based matching. As  $\theta$  increases from 0.5 to 1.5, the accuracy improves from 0.548 to 0.949, indicating that a very small threshold is too strict and may incorrectly reject true anchor-target matches. However, further increasing  $\theta$  reduces the accuracy to 0.827 at  $\theta = 2.0$ , because an overly large threshold may accept dissimilar pairs from different locations. Therefore, we set  $\theta = 1.5$  as the default threshold for the best trade-off between accepting true matches and rejecting mismatched locations.

## 7 Discussion

**Limitation and future work.** Despite promising results, this work suffers from several limitations. First, our data collection is conducted at a relatively small scale and under controlled settings; future work will investigate large-scale, real-world deployments. Second, the current attack relies on GPS labels collected via the same malicious app, which may still raise suspicion among vigilant users. In future work, we will explore alternative ways to obtain such labels from external or auxiliary sources without directly requesting GPS permissions. Third, network and environmental dynamics have not been investigated, as changes in network settings, lighting, or scene layout may alter AR-generated traffic patterns over time; potential techniques such as adaptive or continual learning will be studied in our future work. Fourth, our current approach uses IMU signals primarily for filtering candidate pairs, which may discard true positives, i.e., co-located users with different viewing angles or motion patterns, leading to false negatives. A more advanced direction is to incorporate IMU-based correction into the signal representation for more robust matching. Lastly, scaling the attack to large populations introduces additional storage and computational overhead, particularly for maintaining and matching large volumes of traffic traces; we will investigate efficient indexing, compression, and approximate matching techniques to improve scalability.

**Potential defense.** One potential defense is to introduce controlled randomness or noise into network traffic patterns, such as padding packet sizes or perturbing transmission timing, to obfuscate identifiable patterns. However, this may incur additional bandwidth or latency overhead. Another direction is to enforce stricter access control over sensitive metadata, such as limiting or auditing applications' access to fine-grained network usage statistics. Reducing the availability of such side-channel information can mitigate the feasibility of traffic-based inference attacks.

## 8 Conclusion

In this paper, we present GPSniffer, a group-based location inference attack that exploits encrypted network traffic patterns in AR applications. By leveraging correlations among co-located users, GPSniffer enables location inference without requiring GPS permissions, camera access, or prior environment setup. We design a complete framework that integrates preprocessing, match filtering, and location inference and demonstrate through real-world experiments that GPSniffer achieves high accuracy and outperforms prior

AR-specific attacks without relying on pre-deployed artifacts. Our findings reveal a new privacy risk that even encrypted network traffic in AR systems can leak sensitive information when combined with cross-user correlations. We hope this work raises awareness of such risks and motivates the development of effective defenses to protect user privacy in AR systems and applications.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number 2514616.

## References

- [1] Md Nawab Yousuf Ali, Md Lizur Rahman, and Ifrat Jahan. 2019. Security and privacy awareness: A survey for smartphone user. *Editorial Preface From the Desk of Managing Editor* 10, 9 (2019).
- [2] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. 2009. Surroundsense: mobile phone localization via ambient fingerprinting. In *Proceedings of the 15th annual international conference on Mobile computing and networking*. 261–272.
- [3] Mauro Conti, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. 2015. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security* 11, 1 (2015), 114–125.
- [4] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3, 1 (2013), 1376.
- [5] Neil Zhenqiang Gong and Bin Liu. 2016. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *25th USENIX Security Symposium (USENIX Security 16)*. 979–995.
- [6] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. 2012. Accomplix: Location inference using accelerometers on smartphones. In *2012 Fourth International Conference on Communication Systems and Networks (COM-SNETS 2012)*. IEEE, 1–9.
- [7] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium (USENIX Security 16)*. 1187–1203.
- [8] Yaoqi Jia, Xishu Dong, Zhenkai Liang, and Prateek Saxena. 2014. I know where you've been: Geo-inference attacks via the browser cache. *IEEE Internet Computing* 19, 1 (2014), 44–53.
- [9] John Krumm. 2007. Inference attacks on location tracks. In *International Conference on Pervasive Computing*. Springer, 127–143.
- [10] Weixian Mai and Yinhao Xiao. 2023. A novel GPU based GEO-location inference attack on webgl framework. *High-Confidence Computing* 3, 4 (2023), 100135.
- [11] Gabriel Meyer-Lee, Jiacheng Shang, and Jie Wu. 2018. Location-leaking through network traffic in mobile augmented reality applications. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 1–8.
- [12] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. 2015. PowerSpy: Location tracking using mobile device power analysis. In *24th USENIX Security Symposium (USENIX Security 15)*. 785–800.
- [13] Sashank Narain, Triet D Vo-Huu, Kenneth Block, and Guevara Noubir. 2016. Inferring user routes and locations using zero-permission mobile sensors. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 397–413.
- [14] Arvind Narayanan and Vitaly Shmatikov. 2009. De-anonymizing social networks. In *2009 30th IEEE symposium on security and privacy*. IEEE, 173–187.
- [15] Mirco Rossi, Sebastian Feese, Oliver Amft, Nils Braune, Sandro Martis, and Gerhard Tröster. 2013. AmbientSense: A real-time ambient sound recognition system for smartphones. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 230–235.
- [16] 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)*. 1357–1374.
- [17] Jiacheng Shang, Si Chen, Jie Wu, and Shu Yin. 2020. ARSpy: Breaking location-based multi-player augmented reality application for user location tracking. *IEEE Transactions on Mobile Computing* 21, 2 (2020), 433–447.
- [18] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying location privacy. In *2011 IEEE symposium on security and privacy*. IEEE, 247–262.
- [19] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.
- [20] Tao Wang and Ian Goldberg. 2013. Improved website fingerprinting on tor. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. 201–212.