A Comprehensive Study of Trajectory Forgery and Detection in Location-Based Services

Huaming Yang, *Student Member, IEEE*, Zhongzhou Xia, *Student Member, IEEE*, Jersy Shin, *Student Member, IEEE*, Jingyu Hua, *Member, IEEE*, Yunlong Mao, *Member, IEEE*, and Sheng Zhong, *Senior Member, IEEE*

***Abstract*—Many mobile apps access users' trajectories to provide critical services (e.g., trip tracking). Unfortunately, in such apps, malicious users may upload fake trajectories to cheat providers for illegal benefits. There are few works in the literature that delicately study trajectory forgery problems. In this paper, we first take the perspective of attackers and consider how they would fabricate vivid trajectories confronting a strict provider. In particular, we use the technique of adversarial examples in deep learning to propose a trajectory forgery method, which produces fake trajectories satisfying two conditions: (1) having the motion characteristics indistinguishable from those of real ones, and (2) matching reasonable walking, cycling, or driving routes when being projected to the map. Our experiments show that they can hardly be detected by mainstream trajectory service providers, even after being equipped with machine learning-based approaches. Therefore, we further present dedicated countermeasures by validating the reasonability of reported received signal strength indicator (RSSI) data of scanned WiFi APs in commercial areas and scanned Cellular APs in rural areas, respectively. They can deal well with the most challenging replay scenario, which can hardly be handled by existing radio-based location verification methods. We conduct extensive real-world experiments covering walking, cycling, and driving scenarios to demonstrate the high detection accuracy of both methods.**

***Index Terms*—Trajectory adversarial examples, trajectory forgery attacks, RSSI-based trajectory detection.**

I. Introduction

A LOT of mobile apps access users' GPS trajectories, i.e., time-ordered sequences of GPS positioning results, to provide critical services. For instance, car-hailing apps such as Uber need drivers' GPS trajectories for mileage counting and trip tracking [2]. Kids apps use GPS trajectories to implement so-called geofencing services to ensure the kids do not stray from a route or an area pre-defined by their guardians [3]. Fitness apps such as Keep rely on users' trajectories to calculate and show their jogging statistics (miles, velocities, etc.) [4]. Actually, due to the high value of trajectories, many leading location service providers (LSP, e.g., Baidu [5] and Amap [6] in China) construct open trajectory platforms, which implement general capabilities of trajectory collection, storage, and analysis. App developers can leverage related SDKs and cloud APIs to quickly build their own trajectory-based services and systems.

As the trajectories are uploaded by clients, a potential risk that service providers have to consider is the trajectory forgery at the client side. In many Apps, malicious users have both the motivation and the ability to forge their trajectories. First, such attacks may bring considerable illegal gains. For instance, in car-hailing apps, a malicious driver can upload a fake trajectory to slightly enlarge the counted mileage to increase the revenue. In addition, to encourage more drivers to work on holidays, car-hailing companies may give additional bonuses for each completed deal. It has been reported by recent news that some malicious drivers create false orders and forge driving trajectories with malware to cheat for the bonuses [7]. Second, as the app is running on the devices fully controlled by users, it is not difficult for them to launch trajectory forgery attacks from various ways, e.g., using hooking frameworks (Frida [8], Substrate [9], etc.) at different layers to hook location requesting APIs and manipulate their results. Although there exist some environment-based detection approaches against these hooking frameworks, it is theoretically possible to bypass all of them as a malicious user may have gained the root privilege. In extreme cases, she/he may even install external analog GPS equipment, in which the positioning results are fully controlled.

Therefore, we believe service providers badly call for an effective fake trajectory detection mechanism at the cloud side. Unfortunately, most existing work in this area focuses on validating the authenticity of single points instead of trajectories. Certainly, we can easily extend them to detect forgery trajectories by independently checking every point. As the useful information extracted from the coordinates of a single point is quite limited, all of them seek help from extra geo-dependent data or activities to make the decision. Generally, these schemes can be divided into three categories. The first category [4], [10], [11], [12] deploys dedicated APs in advance at some critical places and then constructs location certification through short-distance communication between users and APs. This type of method is ill-suited for our scenario, in which users' outdoor trajectories may cover an extremely large area (e.g., the driving trajectories of Uber). It is too expensive to deploy so many APs that can cover

Manuscript received 31 August 2022; revised 23 February 2023; accepted 24 April 2023. Date of publication 5 May 2023; date of current version 6 March 2024. This work was supported in part by the NSFC-61972195, the Leading-edge Technology Program of Jiangsu NSF under Grant BK20202001, NSFC-61872179, and NSFC-61872176. Recommended for acceptance by A. Conti. *(Corresponding author: Jingyu Hua.)*

The authors are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: yhmnjucs@163.com; xiazhzh@smail.nju.edu.cn; jersyshin@foxmail.com; huajingyu@nju.edu.cn; maoyl@nju.edu.cn; zhongsheng@nju.edu.cn).

Digital Object Identifier 10.1109/TMC.2023.3273411

the whole area. Rather than dedicated equipment, the second category [13], [14], [15] requires users to certify their locations by performing mutual peer-to-peer communication with another user within a certain distance. Nevertheless, it is hard to guarantee that there are always peers appearing nearby. The last category [16], [17], [18], [19], [20], [21], [22], [23] makes the clients extract some geo-related environment signatures (e.g., signatures of WiFi or Cellular signals) and then compares them with those reported by other users at the same position to find the anomaly. The problem is that the accuracy of the proposed signatures is too coarse, i.e., the range of data variation allowed is too big. As a result, malicious users can escape from being detected by simply replaying their historical data with slight noises. In sumary, none of these existing solutions can meet the needs of trajectory detection in current application scenarios.

Targeting this problem, this paper first takes the perspective of attackers to consider how they will fabricate fake trajectories rather than single fake points. We believe only if figuring out this issue, it is possible to devise a really effective defense scheme. Although some informal web reports say there exists malware that could produce fake GPS trajectories simulating human motions, there is no formal study in the literature. Moreover, the task to generate an indistinguishable fake trajectory is obviously non-trivial as it needs to satisfy at least two requirements. First, as the LSP may extract and check motion features from a given trajectory, the attacker has to guarantee such features of fake trajectories are indistinguishable from those of real ones. Second, when the trajectory is projected to the map, it should briefly match a reasonable walking, cycling, or driving route between the start and the end points.

We therefore propose an outdoor GPS trajectory forgery method that well meets these two requirements. For the motion patterns, considering machine learning-based classification is the most probably validating way used by the provider, we make the attacker also train a classification model (a LSTM-based classifier in our approach [24]) using various public trajectory datasets. Then, a natural approach to meeting the first requirement is to utilize the adversarial examples technique [25] to search for an adversarial trajectory between the given start and the end points that is misclassified as a real one. For the second requirement, we consider two scenarios: the malicious user has a real historical trajectory or not. In the prior case, the best choice is to launch a replay attack, i.e., generating the adversarial trajectory by adding small noises to the historical one. Here, the introduced noises should be small but not too small. It should be small because the historical trajectory is real and must be consistent with the map. If the noises were large, the replayed trajectory might hugely deviate from a reasonable route and thus be detected. Oppositely, if the noises were too small, the replayed trajectory would be too similar to the historical one and also be detected. According to our experiments, even if the same user walks or drives along the same route twice, there must be sufficient difference between the GPS trajectories. We carefully design the loss function to well coordinate these two goals seeming contradictory. For the second case, we leverage an existing navigation service to obtain a recommended route and make sure the generative adversarial trajectory is as close to this route as possible.

Once a fake trajectory satisfies the above two requirements, it obviously becomes extremely difficult for a provider to detect it just based on the trajectory itself. Actually, according to our experiments on the two leading LSPs, none of them can detect our forged trajectories. We then consider a dedicated countermeasure against the proposed attack. It seeks help from the signal features (i.e., RSSI) of nearby WiFi APs, and relies on the fact that it is infeasible for a user to precisely predict the RSSIs of nearby APs at a location that she/he never visited, while it is highly probable for a leading LSP to collect many historical data adjacent to some of the points of a given trajectory due to its huge user base. The provider collects the RSSIs of sensible APs at each location to be validated and then use the crowdsourced historical data to verify the truth of the received RSSIs and further the truth of the trajectory. The major challenge is that if a user has a historical trajectory, she/he could simply replay the corresponding RSSI data by adding slight noises. This requires our verification method should be extremely fine-grained. Existing methods leveraging RSSIs of APs to verify single location truth can hardly deal with the situation [16]. We address this problem from two sides. First, we require a new trajectory that should be sufficiently different from any historical record, which means the replayed locations have to be not too close to the original one and thus there should be considerable variations in real RSSIs. These variations are hard to predict for the user. Second, for each location, our verification method exploits the historical points collected by the LSP within a small neighboring area only and computes the confidence of the reported RSSIs by taking both the density and the distance into consideration. In addition, we use a machine learning approach to integrate the results of all the points of a trajectory to give the final conclusion.

We conducted extensive real-world experiments at three local commercial areas, which cover walking, cycling and driving trajectories. We collected 5,000 trajectories for each area. The results show that our trajectory forgery attack can really escape from the detection of both normal and deep learning-based mobility classification models with a high probability (above 92%). However, these attacks can be well captured (with a probability above 94%) by our WiFi-based countermeasures so long as the average number of reference historical points within a 2.5 m radius is above 4.

Finally, we consider the sparsely-populated suburban areas with few WiFi APs. We extend our defense approach to leverage the RSSIs of 4 G/5 G cellular APs, the radios of which are considered to have covered suburban areas in many countries, such as the US and China, to detect trajectory forgery. Compared with the WiFi-based method, it requires much fewer scannable APs and historical data, which are desired in suburban areas. According to our real-world experiments, as long as there are more than 7 historical reference points on average within a reference radius of 5.6 m for every point ($0.07/m^2$), the detection accuracy of fake trajectories can reach over 85% in all scenarios (walking, cycling, and driving).

II. Machine Learning-Based Trajectory Forgery

Before presenting a trajectory detection scheme, we first consider how would attackers fabricate fake trajectories confronting strict providers. We believe that it is not possible to devise a defense scheme that is really effective until we have a clear understanding about this issue. Unfortunately, there are few practical mechanisms that can offer an explicit way to generate fake trajectories. Therefore, in this section, we focus on proposing a machine learning-based trajectory forgery method that can generate vivid fake trajectories.

A. Security Assumptions & Attack Goal

Trajectory: In this work, trajectory refers to a sequence of positions in continuous time. For the trajectory required by the LSP, the location is the necessary information and some additional information may need to be uploaded (e.g., RSSI, accelerometer, etc.). Trajectories in this section are sequences of $[lat, lon, time]$.

Covert Client-Side Attacks: We assume that the attacker has the ability to make the server obtain fake trajectory information by hooking and modifying local GPS positioning results on the client-side. Additionally, we assume that this process is concealed and cannot be detected on the client-side. Although there do exist some apps that may check their running environments against hooking frameworks (e.g., Frida), this approach is not constantly reliable, especially when the malicious user has gained the root privilege. Theoretically, a privileged user can easily defeat the environment checking codes again because she/he holds at least the same privilege as the app. We do not want to get into this endless arms race on the client-side. So, we assume the provider can only rely on the data uploaded by clients to decide whether a trajectory is real or not.

Outdoor Scene Only: In this paper, we focus on the outdoor scene only. We leave the indoor trajectory forgery and detection in future work.

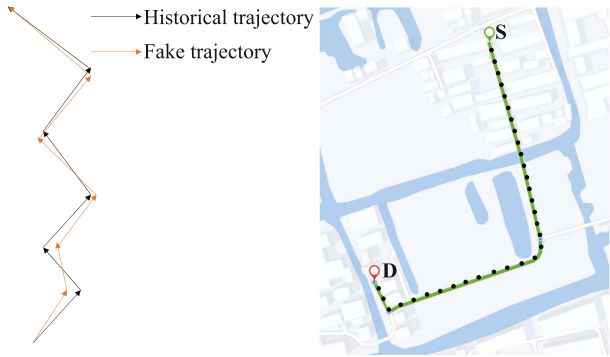
Trajectory Dataset Accessibility: We assume that both attackers and providers can access any open and private trajectory datasets such as GeoLife GPS Trajectories, OpenStreetMap Trajectories[1]. These datasets can be used to learn motion characteristics of humans, and thus are extremely important for both trajectory forgery and detection.

Attack Goal: Given a source S, a destination D and a time sequence $[t_1, t_2, \ldots, t_n]$ where the time interval is a constant c: $c = t_{i+1} - t_i$, our task aims to generate a fake trajectory T=$[P_1, P_2, \ldots, P_n]$, where P_i is the geo-coordinate of the user at time t_i, and $P_1 = S, P_n = D$, and the LSP cannot distinguish it from the a real trajectory between S and D. Here, we define the real trajectories as trajectories produced by real human walking, bike riding, or car driving.

B. Our Proposed Trajectory Forgery Method

To explicate our goal, we first present our definition of indistinguishability for the defender:

[1]OpenStreetMap is a free and open-source platform jointly created by the Internet public [26]. Anyone can download the real trajectory dataset by themselves.



(a) Trajectory replay attacks (b) Attacks based on map navigation

Fig. 1. Examples of our attack: left is our attack with historical trajectories, right is our attack with navigation trajectories.

Motion Characteristics Indistinguishability: A lot of motion characteristics (such as velocity, acceleration, and stopping time) are contained in trajectories and can be discovered and extracted by machine learning techniques such as neural networks. Our forgery method should guarantee the motion characteristics of fake trajectories are indistinguishable from those of real ones.

Route Rationality: A forged trajectory should be consistent with the real-world road system. Specifically, when it is projected to the map, it should briefly match a reasonable walking, cycling, or driving route between the start and the end points.

A natural idea for the LSP to verify a trajectory is to use some machine learning techniques such as neural networks to train a binary classifier based on the motion features of fake and real trajectories. However, recently, many works show that these learning models are vulnerable to adversarial examples. So, we decide to exploit the related technique to generate adversarial trajectories that can escape from the detection of a pre-trained LSTM-based trajectory classification model. Trajectories are time series with strong contextual dependencies. It is impossible to walk from one location to another far away location within a second. LSTM is a neural network for processing sequence data. Compared with the general neural network, it can handle the data of sequence change. Simply put, LSTM can perform better in longer sequences than ordinary RNNs [27].

Since this classification model is built based on a large number of real trajectories, we believe that if a fake trajectory is considered to be normal by this model, its motion features should have been extremely close to those of real ones. For the requirement of route rationality, we guarantee it by strictly restricting the distance from the adversarial trajectory to a pre-determined reference trajectory. Here, a reference trajectory is selected in two ways (Fig. 1). If the user has a historical trajectory between the given points, we directly take it as the reference trajectory. Otherwise, the reference trajectory is generated from a planned route of some navigation service such as amap. Obviously, both of them are rational routes and so long as the generated adversarial trajectory is close to any of them, the requirement of route rationality could be well satisfied. We present our detailed methods for these two scenarios respectively below.

Generate Trajectories Without Historical Records (Navigation Attacks): Given a start point S and an end point D, we

first show how to construct a fake trajectory without historical records.

Since we have to restrict the distance between the generated fake trajectory and the reference one, we need a metric to describe the distance between two trajectories. Dynamic Time Warping (DTW [28]) is a way to measure the similarity between two usually temporal sequences (e.g., trajectories, speech) that do not sync up perfectly. The basic idea of DTW is to find out the warping path W between two trajectories that minimizes the warping cost. We calculate DTW by the following:

$$DTW(T,T') = min\left\{\frac{1}{K}\left[\sum_{k=1}^{K} w_k\right]^{\frac{1}{2}}\right\}, \quad (1)$$

where the w_k is the $k\text{-}th$ element of a warping path. A more detailed introduction is in [29]. Here we employ the function $DTW(T,T')$ to measure the distance between two trajectories.

To conduct our forgery method, we fetch a series of GPS coordinates and an average speed from any commercial navigation system such as Google map. We sample out a trajectory T that matches both the coordinates and the speed.

We present our loss function of constructing a fake trajectory T' from a navigation trajectory T:

$$loss(T,T') = \lambda_1 * loss_{ent}(T,T') + DTW(T,T'), \quad (2)$$

where $loss_{ent}(T,T')$ is the cross-entropy loss function which describes the loss of classification errors and $\lambda_1 > 0$ is a suitably chosen constant. Both $loss_{ent}(T,T')$ and $DTW(T,T')$ are important, (2) not only needs to ensure that the classification is assigned to the specified label, but also needs to minimize the DTW distance to ensure that the forged trajectory is consistent with the real-world road system. However, the magnitude of $loss_{ent}(T,T')$ and $DTW(T,T')$ are different, so we need to use λ_1 as a weight factor.

Generate Trajectories With a Historical Record (Replay Attacks): In this scenario, as the adversary owns a history trajectory between S and D, can she/he simply replay it? It sounds feasible, unfortunately, as the server has the records too, the server can simply traverse its records and differentiate whether the new trajectory is a real one or a replay. Actually, even if the same user walks or drives along the same route twice, there must be sufficient difference between the GPS trajectories.

Therefore, given a historical trajectory T, starting from position S and ending at position D, in order to satisfy our goal, the fake trajectory T' generated by our method should not be too similar with T nor too different from T either.

In order to meet the above requirements, we first define $loss_2$ to describe the compromise of the distance between a fake trajectory and the real one.

$$loss_2(T,T') = max\{DTW(T,T'), \\ 2*(MinD+\delta) - DTW(T,T')\}, \quad (3)$$

where *MinD* is a lower-bound of the distance between any two real trajectories (this threshold is determined through repeated experiments and will be explained later).

For this loss function, on the one hand, we minimize the DTW distance between the historical trajectory T and the fake trajectory T' when their distance is above *MinD*; on the other hand, we constrict the distance to be at least $MinD$ by using $2*(MinD+\delta) - DTW(T,T')$ to prevent the server judging the fake trajectory as a replay of a historical one. At last, as we expect the distance between the fake trajectory and the real one to get close but above $MinD$, we add a small constant δ.

We then present our loss function of constructing a fake trajectory T' from a historical trajectory T:

$$loss(T,T') = \lambda_2 * loss_{ent}(T,T') + loss_2(T,T'), \quad (4)$$

where λ_2 is a suitably chosen constant like λ_1.

Adversarial Trajectory Generation: The C&W algorithm is a classic adversarial example attack algorithm proposed by N. Carlini and D. Wagner [30]. We use the optimization based C&W algorithm to train an optimizer for each original trajectory. The original trajectory T generates a fake trajectory T' in each iteration according to the optimizer. Then we can calculate the $loss$ and make the fake trajectory meeting the requirements by optimizing $loss$. Through continuous iteration, we can find the final solution that can make the loss as small as possible.

III. Forged Trajectory Detection Based on WiFi RSSI Data

Through the fake trajectory generation method we introduced in the last section, we believe it becomes infeasible for the defender to detect the forgery attack based on the trajectories themselves. In this section, we present a dedicated countermeasure by seeking help from WiFi RSSI data. It requires the client to upload WiFi RSSIs of APs around each point while providing the trajectory and then predicts the truth of the trajectory by checking the rationality of these RSSI data.

A. Useful Characteristics of WiFi RSSI

We first show why we choose WiFi RSSI to help detect fake trajectories. WiFi RSSIs demonstrate the WiFi signal strengths of surrounding APs, which show many valuable characteristics that can help to distinguish between a fake trajectory and a real one:

WiFi RSSI is Hard to Forge for Attackers: As WiFi signals could be highly affected by many environmental factors, it is impossible for the attacker to forge RSSIs of WiFi APs at a specific place where she/he never visits.

Even if the attacker has some historical WiFi RSSI of one trajectory, we think a simple replay attack is still challenging. Recall that we require a new trajectory to be sufficiently different from any historical record, which means the replayed location has to keep a considerable distance from the original one and thus there should be considerable variations in real RSSIs in many cases. These variations are still hard to predict for the user.

WiFi RSSI can be Collected in Large by Providers: As we will show below, it is natural for clients of LSPs to gain the privileges necessary for scanning the RSSIs of nearby WiFi APs. Therefore, it is easy for those leading LSPs to collect WiFi RSSI data of a large scale of positions due to its huge user base. The

density of such historical data in some hot commercial areas could be extremely high. Therefore, we have the confidence to assume that within a trajectory to verify (at least in urban areas), there are some (if not all) points with a high probability that contain a certain number of close points whose WiFi RSSI has been recorded by LSPs.

With the above valuable characteristics, the basic idea of our proposal is to leverage the crowdsourced historical data of nearby points (called *reference points* below) to verify the reasonability of the RSSIs reported with the trajectory, the result of which can then further indicate the truth of the trajectory itself. The major challenge is that if a malicious user owns a historical trajectory, she/he could simply replay the corresponding RSSI data by adding slight noises, which requires our verification method to be extremely fine-grained.

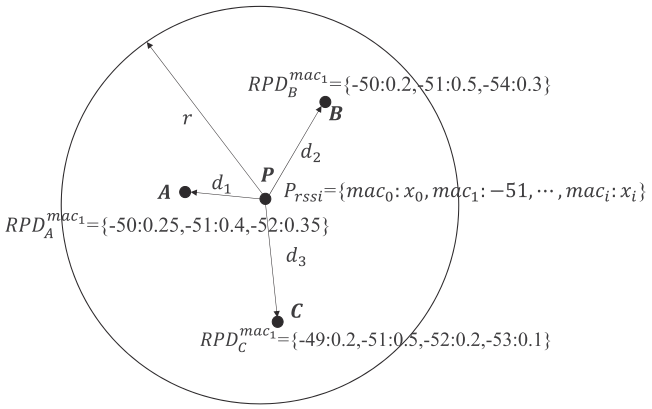


Fig. 2. RSSI verification based on adjacent historical reference points.

B. Security Assumptions & Design Goal

Our proposal has to make two assumptions while being used to verify a trajectory.

Assumption 1: Clients of LSPs have gained the privileges necessary for scanning WiFi RSSIs. On both Android and iOS, an app has to request specific privileges in order to scan nearby WiFi APs and obtain the RSSIs. For instance, Android requires to gain privileges including ACCESS_WIFI_STATE, CHANGE_WIFI_STATE, ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION [31], [32]. The first two are with the prevention level of normal and are naturally requested by almost all the apps to realize dynamic responses to network changes. The prevention levels of the latter two are both dangerous and can be withdrawn by users at runtime. However, they are also two necessary privileges for positioning, which must have been granted by the users for location-based services.

Assumption 2: The average densities of both sensible WiFi APs and referable historical points along the trajectory are not too low: This assumption is obviously necessary as our proposal will use the historical RSSI data collected from adjacent points along the trajectory to verify the reasonability of the RSSIs reported. According to our experiments in Section V-B2, the average number of sensible WiFi APs should be above 8, and the average density of reference points should be above $0.2/m^2$, which we consider being practical at least in most urban areas.

Design Goal: A user uploads a trajectory T=$[P_1, P_2, \ldots, P_n]$ of n points. Here, $P_i = [loc_i, RSSI_i, MAC_i]$ is a triple corresponds to the i-th point, where loc_i are the GPS coordinates, $RSSI_i = [rssi_1, rssi_2, \ldots, rssi_m]$ and MAC_i= $[mac_1, mac_2, \ldots, mac_m]$ are the RSSIs and MACs of m APs scanned at this point, respectively. The time gap between two adjacent points is fixed to t seconds. The provider has employed a crowdsourcing method to collect a RSSI dataset $\mathbb{H} = \{H_1, H_2, \ldots, H_k\}$ from k locations. Here, each data $H_j \in \mathbb{H}$ is a similar triple as $P_i \in T$. Then, our goal is to find a precise prediction function J:$(T, \mathbb{H}) \rightarrow \{0, 1\}$, where 0 indicates T is forged while 1 is opposite.

Focusing on Replay Attacks: Note that in the case of no historical trajectory, WiFi RSSIs can hardly be forged by the attacker as she/he even does not know what APs there are around each point. Attacks, in this case, can be easily detected by simply examining the WiFi RSSIs. Therefore, we only discuss the defense scheme in the case of trajectory replay in this paper.

C. Proposed RSSI-Based Detection Method

In this part, we present our fake trajectory detection scheme based on WiFi RSSI data. It requires each mobile phone user to provide the WiFi RSSI data at each point of the trajectory. As Fig. 2 shows, for each point $P \in T$, our scheme tries to leverage the RSSIs of those historical points within a circle of radius r around P to estimate the confidences of its reported RSSI values. We call such points *reference points* and the circle $C_P(r)$ *the reference area*. The *confidence* here refers to the probability that a specific RSSI value is considered to be really true. Our intuitive thought is that if r is small, the RSSI values of P should be close to those of the reference points. However, "close" does not mean to be completely identical. In fact, we believe it is impractical to predict the exact differences between in our scenario due to two reasons.

First, due to GPS errors, we actually do not know the exact positions of both P and the reference points. Second, even if we know their exact positions, the RSSI of an AP at a specific position is chaotic to some extent and heavily affected by the environment and the receiving device itself. Considering these issues, our scheme does not aim to employ any theoretical signal attenuation functions, which usually rely on the precise positions of equipment and can only work well under ideal conditions to predict such differences. Instead, we simply regard the RSSIs of an AP nearby a reference point as a discrete random variable within a specific interval. Our scheme then tries to estimate the probability distribution of this variable and directly takes the probability of the reported RSSI according to this distribution as the confidence estimation from the reference point. The details are shown below.

RSSI Probability Distributions (RPDs) Around Historical Points: For each historical point H in the provider's dataset $\mathbb{H}$, we define a neighboring area $C_H(R)$, which is a circle of radius R around H. As mentioned above, our scheme regards the RSSIs of a certain AP mac_i within $C_H(R)$ as a random variable. We estimate its probability distribution based on all the historical

points within $C_H(R)$. In particular, for a possible RSSI value x of AP mac_i, the estimated RPD function is

$$RPD_H^{mac_i}(x) = \frac{\|\{Q \in \mathbb{H} | Q_{rssi_i} = x \wedge Q \in C_H(R)\}\|}{\|C_H(R)\|}, \tag{5}$$

which is just identical to the ratio of historical points having the specific RSSI value x within this area. In other words, our basic idea is to the use the frequencies of RSSI values to approximate the probability distribution. Denote by $\mathbb{R}$ the support of the RSSI values of the AP mac_i at all the historical points in $C_H(R)$. Obviously, $\sum_{x \in \mathbb{R}} RPD_H^{mac_i}(x) = 1$. In other words, for all possible x, $RPD_H^{mac_i}(x)$ integrates to one. With this distribution, the estimated confidence of P_{rssi_i}, the reported RSSI of AP mac_i at position P, according to the reference point H is $RPD_H^{mac_i}(P_{rssi_i})$.

Because there might be more than one reference point in $C_P(r)$, we have to integrate all their confidence estimations to obtain the final confidence about each reported RSSI. In this process, we consider the following two factors to assign different weights to individual estimations:

Distance of a Reference Point: Obviously, a closer reference point should play a more important role in the RSSI verification. Therefore, we introduce a weight parameter $\theta_1(H, P)$ to consider this fact in the final RSSI confidence calculation: given a reference point $H \in \mathbb{H}$,

$$\theta_1(H, P) = \frac{\frac{1}{d_{euc}(H,P)}}{\sum_{H \in C_P(r)} \frac{1}{d_{euc}(H,P)}}, \tag{6}$$

where $d_{euc}(H, P)$ is the euclidean distance between H and P.

Reliability of Estimated RPDs: Besides the distance, the reliability of the estimated RPD values could also significantly affect the accuracy of such confidence calculation. Obviously, a reference point with more points in its RPD counting area, the obtained RPD statistics are more reliable. In other words, the reliability of RPD is highly related to the density of the counting area. So we also define another weight $\theta_2(H)$ to consider this factor for each reference point H:

$$\theta_2(H) = 1 - \left(\frac{1}{t}\right)^{\epsilon}, \tag{7}$$

where ϵ is the density, and

$$\epsilon = \frac{\|C_H(R)\|}{\pi R^2},$$

and t is a variable used to constrain the density within [0,1]. We set $\frac{1}{t} = 0.9$ for convenience.

WiFi RSSI Confidence Calculation: Considering the above factors, we can finally calculate the confidence of a reported RSSI value $rssi_i$ of AP mac_i at position P:

$$\Phi_P(P_{rssi_i}) = \sum_{H \in C_P(r) \wedge \mathbb{H}} \theta_1(H, P) * \theta_2(H) * RPD_H^{mac_i}(P_{rssi_i}). \tag{8}$$

The larger this value is, the more confident the RSSI value uploaded by the user receiving mac_i at position P is.

Forgery Trajectory Detection: Based on the above, the confidence of individual RSSI values can be roughly estimated now. We finally integrate all the estimated confidences values of all the RSSI values in a trajectory as the features to predict the truth of the trajectory based on a machine learning approach.

To use the machine learning model, we need a fixed-length eigenvector for subsequent training and prediction. In other words, we have to first determine the feature vector. At each location, we take the k strongest WiFi RSSIs into consideration as the RSSI values of weak signals are less accurate and may fluctuate badly. The value of k is experimentally determined. Then, the features we collect for a single point P_j in the trajectory T is

$$feat_j = [(Num_{P_{j_{mac_1}}}, \Phi_{P_j}(P_{j_{rssi_1}})), \ldots, (Num_{P_{j_{mac_k}}}, \Phi_{P_j}(P_{j_{rssi_k}}))],$$

where Num_{mac} represents the total number of reference points used for calculating the RSSI confidence of AP mac. Obviously, the more reference points are used, the more accurate the estimated confidence is. Hence, we include Num_{mac} as a key feature to indicate how many weights need to be assigned to each aggregated RSSI confidence in the final trajectory verification. The other implicit number is the number of historical points used to count the RPD around each reference point. We incorporate it as the density of the RPD counting area (i.e., $\theta_2(H)$ in (8) to measure the reliability of the estimated RPD itself. Once again, we believe that a reference point with more neighboring historical points in its RPD counting area yields more reliable RPD statistics. So, the motivations to incorporate these two numbers are also totally different.

So, the final feature vector for the whole trajectory is

$$feature = [feat_1, feat_2, \ldots, feat_n]. \tag{9}$$

We finally train an XGBoost-based [33] binary trajectory classification model with this feature vector to detect fake trajectories. For this purpose, as we mentioned in Section V, we did real-world experiments to build a RSSI training dataset in three local commercial areas covering walking, cycling and driving scenarios. In each area, we collected RSSI data from more than 50,000 points. The evaluation results demonstrate that our proposal could achieve a detection accuracy above 90% even when the average density of reference points is just $0.2/m^2$, which could be satisfied in most downtown areas.

Experimentally Determine R.*R* is the radius when calculating $RPD_H^{mac_i}(x)$. If R is too small, then there are not enough points to calculate $RPD_H^{mac_i}(x)$. If R is too large, it will contain many irrelevant points, which will have a bad effect on $RPD_H^{mac_i}(x)$.

To measure *R*, we collect 1,000 GPS samples at the same location and use the sample center point P as the real coordinate. For each sample, we place the GPS location P' in a 2D coordinate system with the true location P as the original, with the positive X-axis pointing East and the positive Y-axis pointing North. We then calculate the polar coordinates $P' = (d, \alpha)$ of each sample, where d and α represents the length and the angle of the vector $\overrightarrow{PP'}$ in its coordinate system. In particular, the statistical results of this experiment show that the angle α is

approximately uniformly distributed in $[0, \pi]$ and the length d is approximately right-half normal distributed $N(0, \sigma^2)$ where $d > 0$. According to the 3 sigma rule of the normal distribution, 99% of the sample points are distributed within 3σ of the center point where $\sigma = 0.5\,m$. Therefore, we define the maximum distance of deviation $R = 6\sigma = 3$ m.

IV. Proposed Cellular RSSI-Based Detection Method

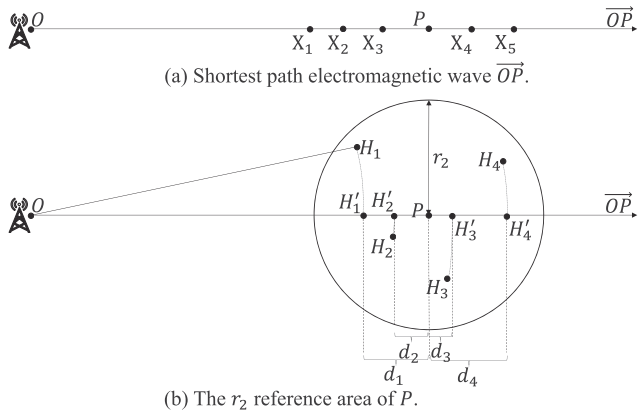
The trajectory forgery detection method proposed in the last section relies on a certain density of WiFi signals. Unfortunately, in many sparsely-populated suburban areas, there are usually much fewer and even no WiFi networks, and thus, this method becomes no longer applicable. Compared with WiFi, 4 G/5 G cellular networks are obviously much more widely available due to their signals' longer propagation distances. In many countries, such as the US and China, the 4 G/5 G networks have covered even those rural areas. So, a natural idea is to use RSSIs of cellular APs rather than WiFi APs in these areas to perform the trajectory validation. Nevertheless, it confronts at least the following two challenges.

Challenge 1: Low density of APs: In our RSSI-based detection method, the number of APs that a mobile device could scan (i.e., the AP density) at a location plays an important role. Unfortunately, communication operators often do not deploy APs intensively to reduce costs, especially in suburban districts. Correspondingly, the Android API only scans the information of at most 4 cellular APs every time by default.

Challenge 2: Low density of historical reference points: Besides the density of APs, the density of historical reference points is another critical factor in our method. According to our experiments in Section V-B2, the density of WiFi reference APs should reach $0.2/m^2$ to guarantee a high detection accuracy. However, the density of reference historical points in sparsely-populated rural areas is usually much lower.

Due to the above problems, if we directly apply the WiFi-based approach in the last section to cellular RSSI data, its performance would be far from satisfactory. We conducted a real-world experiment in a walking scenario with a $0.09/m^2$ density of historical reference points. The test set contains 1,000 real trajectories and 1,000 fake trajectories. The results show that the detection accuracy can only reach 69.4%, which is much lower than the results using WiFi data in Section V-B2. So, we need a more refined and dedicated method while leveraging cellular RSSI data.

In this new method, we first assume the provider can roughly estimate the location of each scanned AP using its RSSIs collected from historical data. We will defer the positioning method at the end of this section. As shown in Fig. 3(a), there is a position point P in the trajectory to be validated, which provides the signal strength P_{rssi_O} with respect to cellular AP O. The basic idea of our new detection method is to consider the straight-line (i.e., the shortest) signal propagation path $\overrightarrow{OP}$ from O to P. Suppose there are some crowdsourced historical points $\mathbb{X} = \{X_1, X_2, \ldots, X_t\}$ that happened to be on this path, then the corresponding RSSI should decrease along the $\overrightarrow{OP}$. In this way, we can infer the authenticity of P_{rssi_O} based on the positional relationship between point P and $\mathbb{X}$.



(a) Shortest path electromagnetic wave $\overrightarrow{OP}$.

(b) The r_2 reference area of P.

Fig. 3. Cell verification based on adjacent historical reference points.

However, due to the low density of historical points in suburban areas, it is unlikely that any reference points happened to be on $\overrightarrow{OP}$. So, we turn to the reference points within the reference area (circle $C_P(r_2)$) of distance r_2 from point P. Given a reference point $H \in \mathbb{H}$, we can find a corresponding point H' on $\overrightarrow{OP}$, where $d_{euc}(O, H) = d_{euc}(O, H')$ (Fig. 3(b)). Because $r_2 \ll d_{euc}(O, H)$, and the environment in suburban districts is simple, we think that $loss(O, H) \approx loss(O, H')$. So

$$\left.\begin{array}{c} d_{euc}(O,H) = d_{euc}(O,H') \\ loss(O,H) \approx loss(O,H') \\ r_2 \ll d_{euc}(O,H) \end{array}\right\} \Rightarrow H_{rssi_O} \approx H'_{rssi_O}. \quad (10)$$

Based on this inference, for all $H \in C_P(r_2) \wedge \mathbb{H}$, we define $\mathbb{H}' = \{H'\}$ and intuitively estimate P_{rssi_O} by referring to $\mathbb{H}'$ on $\overrightarrow{OP}$. We require that in the most extreme case, there are at least 2 historical points around the valid location point P used for detection in the trajectory ($|\mathbb{H}'| \geq 2$).

RSSI of $\mathbb{H}'$ Should be Decrementing Along the $\overrightarrow{OP}$: $\mathbb{H}'$ should decrease along the op direction like $\mathbb{X}$. Otherwise, the point P is wrong. For $\mathbb{H}'$, if $\exists H_a, H_b \in \mathbb{H}' \wedge d_{edu}(O, H_a) < d_{edu}(O, H_b)$, this condition can formally described as

$$\Gamma(\mathbb{H}') = \begin{cases} 0 & \text{if } H_a.rssi_O < H_b.rssi_O \\ 1 & \text{else} \end{cases}. \quad (11)$$

RSSI Estimation: Even if the RSSI of $\mathbb{H}'$ meets the requirements in descending order, P_{rssi_O} may be forged. Here, we estimate the RSSI of the P where the fake location is not detected, and then compare $E(P_{rssi_O})$ with the P_{rssi_O} provided by the user. If it is within the allowable error range, it is considered to be true, otherwise it is a fake RSSI. We formulate the estimation scheme as follows.

$$E(P_{rssi_O}) = \sum_{H' \in \mathbb{H}'} \frac{\frac{1}{|d_{euc}(H',O) - d_{euc}(P,O)|}}{\sum_{Z \in \mathbb{H}'} \frac{1}{|d_{euc}(Z,O) - d_{euc}(P,O)|}} * H'_{rssi_O}. \quad (12)$$

Confidence: Based on the estimated RSSI value $E(P_{rssi_O})$ from historical signal points, we first calculate the deviation

Δ_O between the user-reported value P_{rssi_O} and $E(P_{rssiO})$, i.e., $\Delta_O = |E(P_{rssiO}) - P_{rssiO}|$. We then use this deviation to determine the authenticity confidence of P_{rssi_O} as follows:

$$\phi(P_{rssiO}) = \Gamma(\mathbb{H}') * \left(1 - \frac{1}{1 + e^{-\Delta_O}}\right). \tag{13}$$

Note that as $E(P_{rssiO})$ may contain errors, the relation between the deviation Δ_O and the authenticity confidence of P_{rssiO} is not straightforward. To address this issue, we conducted real-world experiments as Section V-C shows to collect the actual RSSIs of a large number of historical points. Then, we divided the domain of Δ_O into small, continuous intervals and counted the ratio of the actual historical data points, for which the errors in the estimated RSSI values (using our method above) fell into each interval. We believe that this ratio can be an excellent estimate of the authenticity confidence of P_{rssiO} for a specific Δ_O. Furthermore, our results show that the function $f(\Delta_O) = 1 - \frac{1}{1+e^{-\Delta_O}}$ can ideally model such relations. In addition, since we define an edge case in (11) above, we multiply $\Gamma(\mathbb{H}')$ on the left side to ensure that this error case is ruled out.

In the WiFi setting, we actually use frequency approximation to determine the probability distribution of RSSI based on historical points near the user's location (8). This enables us to calculate the confidence level in the authenticity of the user-reported RSSI. However, in situations where historical data is scarce, using frequency approximation can lead to significant errors. In contrast, cellular signals offer broader and more stable coverage, which allows us to estimate RSSI based on fewer historical points. So, we evaluate the confidence by comparing the estimated RSSI with the user-provided RSSI (13).

Cellular APs Localization: Communication operators deploy Cellular APs to provide communication services. To reduce costs, the locations of APs will be planned to avoid excessive concentration. For example, the Android API defaults to scanning the RSSI of up to 4 cellular APs at a time in one location. In an ideal situation, the electromagnetic waves emitted by the AP encounter the same propagation medium and loss in all directions, so the RSSI of the AP radiation area decreases outward in a ring shape. Based on this inference, we fit the distribution of historical signal point sets of the same RSSI into a circle and then find the circle's center O_s respectively. Finally, we estimate the AP location according to the weighted average of the number of historical points corresponding to different RSSIs.

For RSSI $s \in [-113, -51]$,[2] there are N_s history points. We obtain the recorded RSSI set $\mathbb{S}$ and the set $\mathbb{N}$ of the corresponding number of historical points by looking for the historical points with $N_s \geq 3$. We calculate $\mathbb{O} = \{O_s | s \in \mathbb{S}\}$ and the location of the cellular AP as follows.

$$Loc_O = \sum_{s \in \mathbb{S}} \frac{N_s}{\sum \mathbb{N}} * O_s. \tag{14}$$

As shown in the Fig. 4, for the historical signal points of $rssi_{\mathbb{P}} = -67$ and $rssi_{\mathbb{Q}} = -66$, find the circle centers $O_{\mathbb{P}}$ and $O_{\mathbb{Q}}$, respectively. The position of O is then estimated based on N_{-67} and N_{-66} weights.

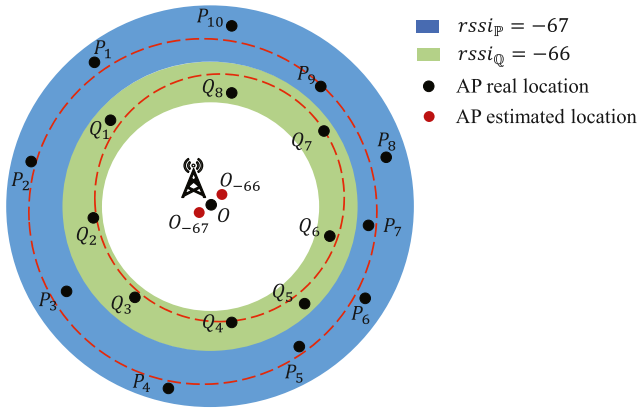


Fig. 4. For $\mathbb{P} = \{P_1, P_2, \ldots, P_{10}\}$ with $rssi_{\mathbb{P}} = -67$ and $\mathbb{Q} = \{Q_1, Q_2, \ldots, Q_8\}$ with $rssi_{\mathbb{Q}} = -66$, fit the distributions of $\mathbb{P}$ and $\mathbb{Q}$ to a circle, respectively, and find the estimated positions $O_{\mathbb{P}}$ and $O_{\mathbb{Q}}$.

We conducted 1,000 positioning experiments on 10 cellular APs in the real scene, and the results show that the average positioning error is 15 m. In the real scene, we think this is an acceptable positioning error. For example, we found the coordinate O of an AP, and then we made 1,000 RSSI measurements over an area with an average distance of 150 m from AP O. After modifying the real position of AP O to O' within 15 m, we used O point and O' to estimate RSSI respectively, and $\Delta(rssi) = |rssi_O - rssi_{O'}| < 0.01$, which has almost no effect on the final classification results.

Forgery Trajectory Detection: We also use machine learning to train the detection model because of the similarity between Cellular RSSI and WiFi RSSI. The detection method is consistent with WiFi RSSI-based detection that first extracts features and then uses Xgboost to train a binary detection model.

Features have some differences from Section III-C. We will ask the user to upload 4 cellular RSSI messages at each location. Using eci to differentiate RSSI, the features we collect for a single point P_j in the trajectory T is

$$feat_j = [(Num_{P_j}.eci_1, \phi(P_j.rssi_1)), \ldots, (Num_{P_j}.eci_4, \phi(P_j.rssi_4))]$$

where $Num_{P_j}.eci_i$ represents the total number of reference points used for calculating the RSSI confidence of AP eci_i. We include $Num_{P_j}.eci_i$ as a feature as the more reference points are used, the more accurate the estimated confidence should be.

The final feature vector of the entire trajectory is consistent with WiFi RSSI-based detection (9). The training process is similar to WiFi RSSI-based detection, and we conduct real-world experiments to construct an RSSI training dataset in local suburban areas, covering walking, cycling, and driving scenarios. In each scenario, we collected RSSI data from more than 50,000 points. The evaluation results demonstrate our proposal could achieve a detection accuracy above 85% even when the average density of reference points is just $0.07/m^2$, which could be satisfied in most rural areas.

[2][-113, -51] is the value range of getRssi() given by Android API.

V. Experiments and Results

We propose a trajectory forgery method based on machine learning and a dedicated countermeasure based on WiFi signal strength. In this section, we will evaluate these two methods through experiments.

A. Evaluation of the Proposed Trajectory Forgery Attack

In our experiments, we first train a LSTM-based classifier C against naive attacks under two datasets. Then, we use classifier C as the target model to conduct adversarial examples attacks in two scenarios (Replay and Navigation) to generate the fake trajectories. Finally, to evaluate the transferability of our attack, we train another three different classifiers against naive attacks and check whether our fake trajectories can fool them, too.

1) Datasets: We use the following two datasets to train the target detection models (i.e., binary classifiers) in the replay and the navigation scenarios, respectively.

OSM. As stated in Section II-A, both attackers and providers can access openstreetmap Trajectories. We downloaded the trajectories within two months from June 2020 to August 2020 on the Openstreetmap website as the real dataset. Because the real trajectory is irregular, we use 1 s as the time interval and select 400 consecutive position points as real trajectory data. After preprocessing, we get a dataset of 50,000 real trajectories named OSM.

AN. Suppose a malicious user needs to forge a trajectory that has not been visited. In that case, she/he can use map navigation to plan a route by specifying the starting position S and the ending position D. We choose Amap navigation to generate fake trajectories in our experiments [6]. We randomly selected 10,000 location pairs in Nanjing, China, and planned the route between each pair using walking, cycling, and driving, respectively. According to the route feedback from Amap, we set a reasonable speed. We then sample at 1 s intervals on the route based on this speed. After randomly selecting 400 consecutive position sequences as fake trajectories, we obtained a fake dataset AN containing 30,000 trajectories.

2) Target Model: It is difficult to know the classification model (model architecture and parameters) used by the server to generate the adversarial trajectory. In our work, we assume that the attacker does not know the specific detection method of LSPs, and only has one or more real trajectory data sets collected by himself or made public. LSTM is usually used for vehicle driving prediction, natural language processing, text matching, and other problems. We employ LSTM to generate our target classifier and then conduct an adversarial example attack on this LSTM classifier.

Naive Attacks: Both OSM and AN are easily obtained by attackers. In the replay scenario, a naive attack in our experiments simply replays an existing trajectory in OSM by adding a tiny noise, which follows the normal distribution $N(0, 0.25)$. This distribution is obtained according to the experimentally measured GPS error distribution, which is described at the end of Section III-C. In the navigation scenario, to avoid being directly detected by the defender through the direction of displacement per second, the trajectories in AN also need to perform naive attacks.

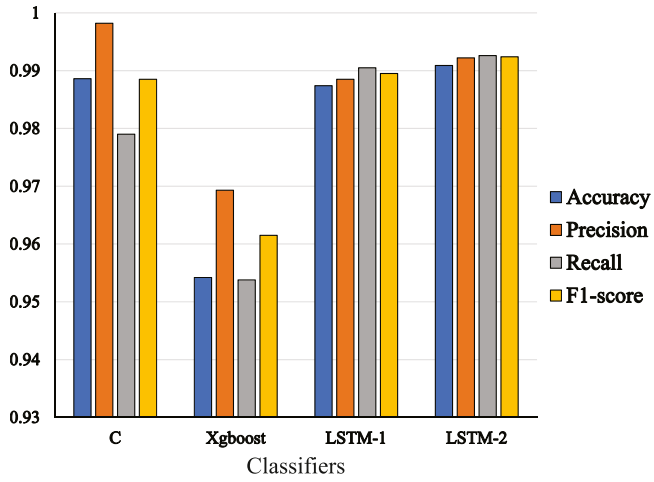


Fig. 5. Classification performance against naive attacks. Model C is the target model of trajectory adversarial examples attacks.

We use the above naive attacks to generate 10,000 fake trajectories for each scenario. Then, we randomly select 10,000 from them and 20,000 from OSM (i.e., the real trajectories) to form the labeled training set D_{train}. The remained 10,000 and another 10,000 real trajectories from OSM form the test set D_{test}. For the trajectory $T{=}[P_1, P_2, \ldots, P_n]$, the displacement between two adjacent points is denoted as $\Delta(P_i, P_{i+1}) = (Edu(P_i, P_{i+1}), Angle(P_i, P_{i+1}))$. Here, $Edu(P_i, P_{i+1})$ represents the euclidean distance, and $Angle(P_i, P_{i+1})$ represents the direction of this displacement. We set the LSTM input layer size to 798, the LSTM hidden layer size to 256, and use Sigmoid function to activate it. We set the learning rate to 0.001, and use the cross-entropy function as the loss. The training is performed 100 rounds in total and the detection results on the test set are shown in Fig. 5, which are extremely accurate.

3) Evaluation of the Proposed Forgery Method: When we apply C&W [30] attacks to classification network C, there are some parameters that need to be determined through experiments.

Choosing the Number of Iterations: The value of iterations is an important parameter. If the number of iterations is too large, it will bring a considerable time cost. If the number of iterations is too small, it may not be possible to find the adversarial examples or deviate from the road system. We use (2) as the loss function for a navigation trajectory T in AN and then use the C&W attack to generate adversarial examples of T. We set the number of iterations to 5,000, and the relationship between the number of iterations and time cost and distance is shown in Fig. 6.

When the number of iterations$\leq$400, no adversarial examples can be found. Afterward, the adversarial examples can be found, and $DTW(T, T')$ drops rapidly. When the number of iterations$>$1,500, the downward trend of $DTW(T, T')$ becomes slower and slower. At the same time, the iteration time on the CPU and GPU increases as the number of iterations increases. Therefore, we chose to perform 1,500 iterations on the trajectory.

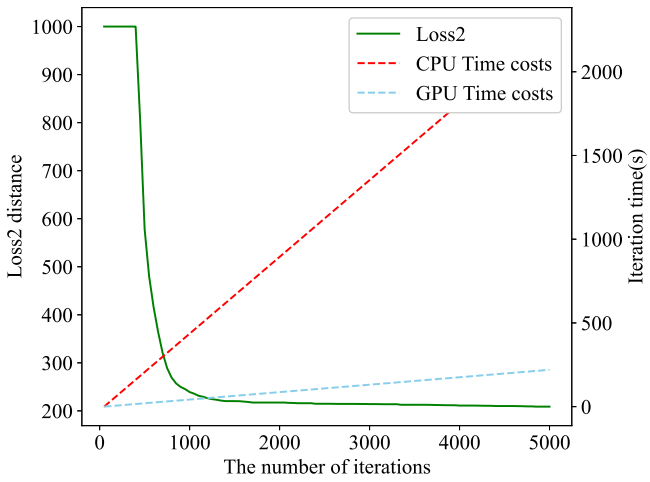


Fig. 6. Variation curve with the number of iterations.

TABLE I
SUCCESSFUL DETECTION RATE AGAINST ADVERSARIAL ATTACKS

Models	Replay attacks	Navigation attacks
C(LSTM)	0.0%	0.0%
XGBoost	4.7%	3.3%
LSTM-1	7.5%	6.8%
LSTM-2	7.4%	7.6%

Experimentally Determine $MinD$: For the replay attack scenario, we need to determine the minimum threshold $MinD$ of the distance. There are also differences between the trajectories formed by the same person and the same equipment taking the same route multiple times. Here, we walked a 200 m route continuously 50 times. The results show that this threshold $MinD_1$ for walk exists and its value is $MinD_1$=1.2/m. Similarly, we experimented with cycling and driving scenarios and got $MinD_2$=1.5/m, $MinD_3$=1.4/m.

After setting the number of iterations to 1,500 and setting the parameters λ (both λ_1 in (2) and λ_2 in (4)) to be automatically adjusted, we forged 1,000 fake trajectories in each of the two scenarios. All the generated fake trajectories can well escape from the detection of C (Table I).

In our attack experiment, the attacker just uses the LSTM model trained based on these public datasets as a detection model to generate fake trajectories whose motion features are as indistinguishable from real trajectories as possible, rather than assuming that LSP must use this classification model.

4) Transferability Evaluation: We further train other completely unrelated detection models to verify the transferability of the attack [34].

XGBoost. The first detection scheme uses the classic XGBoost algorithm. For each trajectory, we do the following feature extraction:

- *Location feature*: Start position, end position, start time, end time.
- *State feature*: The speed and acceleration of the trajectory, the speed and acceleration in the longitude direction, the speed and acceleration in the latitude direction, Velocity difference in longitude and latitude.

We use D_{train} from Section V-A2 as the training set. We perform feature extraction on D_{train}, using a learning rate of 0.0003. The performance of XGBoost classifier against naive attacks is shown in Fig. 5.

LSTM-1: Classifier C uses displacement $\Delta(P_i, P_{i+1}) = (Edu(P_i, P_{i+1}), Angle(P_i, P_{i+1}))$ to describe a trajectory. Here we use $\Delta(P_i, P_{i+1}) = (x_{i+1} - x_i, y_{i+1} - y_i)$ to describe a trajectory and retrain an irrelevant model LSTM-1. The performance of LSTM-1 against naive attacks is shown in Fig. 5.

LSTM-2: We modify the structure of network C, add a hidden layer of size 256, and perform 100 rounds of training using the same scale data set to obtain the model LSTM-2 [35]. The performance of LSTM-2 against naive attacks is shown in Fig. 5.

Finally, we select 1,000 real trajectories from D_{train} for replay attacks and 1,000 fake trajectories from AN for navigation attacks (Section V-A3). Then we use three detection models to detect these 2,000 trajectories and count the number of successfully detected fake trajectories. The rate of successfully detected fake trajectories is shown in Table I. The results show that our forgery scheme has good transferability.

To evaluate the real performance of the proposed attack against LSPs, we searched for some leading commercial SDKs that provide trajectory services (e.g., the Hawk-eye tool of Baidu Map) and tested to see if anything unusual would happen when they are fed fake trajectories generated by our method. Unfortunately, experiments show that they fully accepted all the fake trajectories. Experiments prove that our attack is effective. Trajectory adversarial example attacks are highly transferable and can generate indistinguishable fake trajectories. It is impossible to effectively defend against such attacks only through the location information of the trajectories.

B. Evaluation of WiFi RSSI-Based Forgery Detection

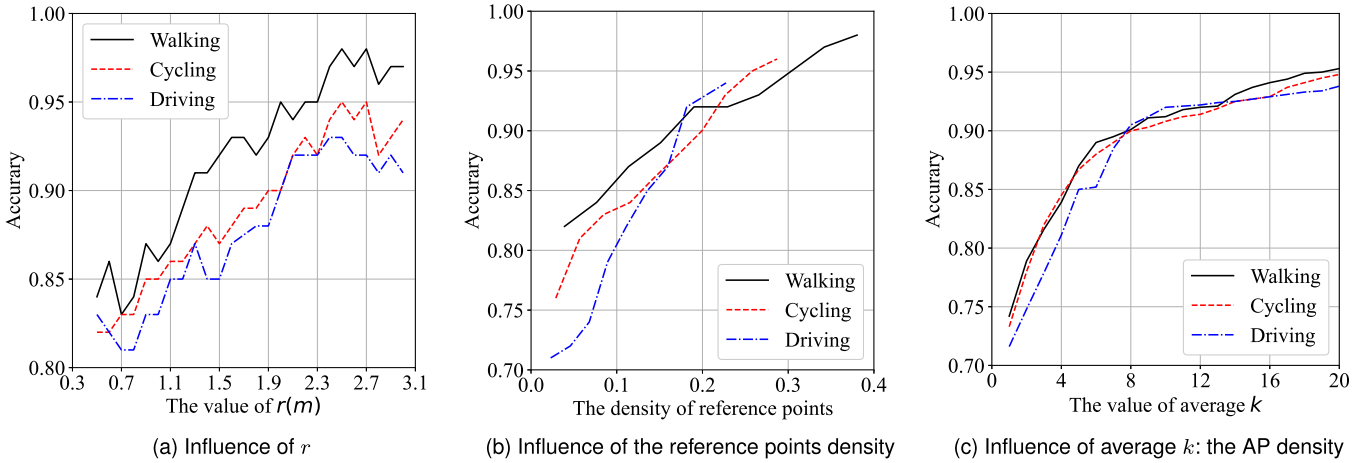
WiFi fingerprints are often used for indoor positioning, and there is little research on outdoor WiFi. Therefore, no public data set can be used, and the attacker cannot use some heuristic rules to add noise to the signal strength. We wrote a signal collection application and collected the trajectories of three modes of transportation, including walking, cycling, and driving.

1) Datasets: The datasets we collected are as follows.

Walking. The walking dataset is collected from the out door area of a large shopping mall. It contains 5,000 one-minute walking trajectories of volunteers within a month. Each trajectory contains 30 position points, and the sampling interval is 2 s. The sampling area of A is 3.4 hm^2. We collected a total of 1,665,264 signal strength records of 5,602 APs.

Cycling: Street B is a pedestrian street downstairs in the community, and many office workers pass by here every day. Using the same time interval, we collected 5,000 one-minute cycling trajectories of volunteers here. Each one also contains 30 points. The sampling area of B is 4.1 hm^2. We collected a total of 1,466,167 signal strength records of 6,567 APs.

Driving: Road C is a main road in a commercial area of our city. A large number of vehicles pass by here every day. We collected the driving trajectories here. Using the same sampling interval, the volunteers collected 5,000 trajectories, where each



(a) Influence of r (b) Influence of the reference points density (c) Influence of average k: the AP density

Fig. 7. WiFi RSSI-based detection experiments.

TABLE II
THE STATISTICAL INFORMATION OF k

	Walking	Cycling	Driving
Average k	29	26	9
Minimial k	3	5	0
90% points	$k \geq 14$	$k \geq 15$	$k \geq 4$

one still contains 30 location points. The sampling area is 5.9 hm^2. We collected a total of 517,526 signal strength records of 6,219 APs.

Denote by k the number of APs received by the user at each location. The statistical information of k is shown in Table II.

For each scenario, we randomly select 4,000 trajectories of 5,000 as the historical data kept by the providers. The RSSI data of the points in these trajectories will be utilized to judge the truth of the newly uploaded trajectories. In addition, we also use these trajectories to build the training set for training the XGBoost-based binary classifier introduced in Section III-C. In particular, we first randomly select 3,000 from them to serve as the normal trajectories. Then, we perform two trajectory forgery attacks introduced in Section II-B with each of the remained 1,000 trajectories to produce extra 2,000 fake trajectories. The RSSI data of them are generated by adding a disturbance randomly selected from three values $\{-1, 0, 1\}$ to their original values. These 5,000 trajectories form the final training set. The test set in each scenario is also composed of 1,000 real trajectories and 1,000 fake ones. The real ones are just the remaining 1,000 non-historical trajectories. The fakes ones are generated from 1,000 randomly-selected historical trajectories with the same method used to produce fake samples in the training set.

2) WiFi RSSI Based Detection and Results: There are some parameters that will have a certain impact on the experiments, and we evaluate them one by one.

The Influence of Reference Radius r. To determine the effect of the value of r on the experimental results, we observe the changes in the accuracy of the detection model by modifying r. The results are shown in Fig. 7(a). When $r < 1\,m$, because r is relatively small and there are not enough reference points, the accuracy is easily affected by individual points, which makes the accuracy change irregularly or even drops as r increases. When $r > 1\,m$, due to the expansion of the reference area, the number of points available for reference is also increasing, the accuracy rate continues to rise and the maximum value is obtained when $r = 2.5\,m$. When $r > 2.5\,m$, some unimportant points are referenced so that the accuracy rate does not increase or even decrease.

The Influence on Density of Reference Points: The average density of reference points within the reference area is another important factor affecting the detection accuracy. If there is no certain number of reference points, even if there is a lot of WiFi information near the track, the server still cannot judge the authenticity. We define density as the average number of reference points per square meter in the reference area of each trajectory point. We modify the density by randomly deleting a portion of reference points to observe the inaccuracy changes. The result is shown in Fig. 7(b).

It shows that the accuracy of forgery trajectory detection in each region increases with the density of reference points around the trajectory. When the density is greater than $0.2/m^2$, which we think could be satisfied in most downtown areas, the detection accuracy is greater than 90%.

The Influence of Average k: The average number of APs in the RSSI information submitted by users will affect the detection results. In some areas, no stores emit WiFi signals. To verify the robustness of our detection scheme, we designed an AP density experiment. For each trajectory, we change k by randomly deleting some APs, and then use the change of k to observe the detection results, which are shown in Fig. 7(c). As the average k increases, the detection accuracy continues to rise. After the Driving reaches the average k value of the entire data set, the rising speed is significantly reduced, and the final detection result is lower than that of Walking and Cycling. In the extreme case of $k = 1$, the detection accuracy of more than 70% is still maintained in all the three scenarios. When the average $k > 7.5$, the detection accuracy of all trajectories is above 90%, which demonstrate our proposal is really effective

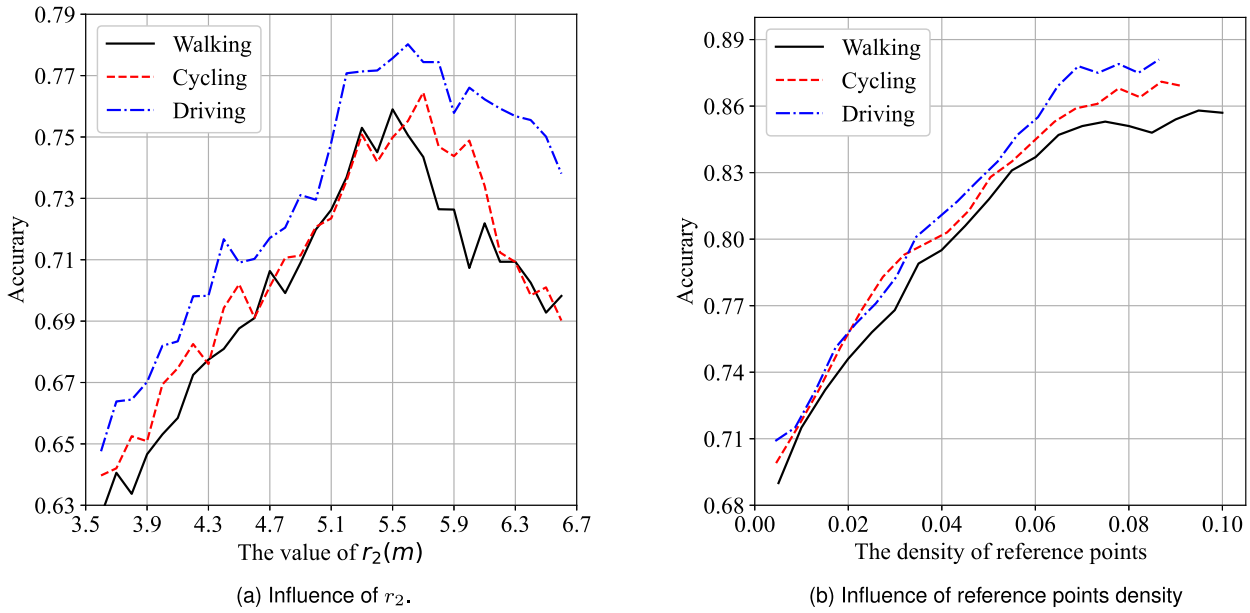


Fig. 8. Cellular RSSI-based detection experiments.

in most commercial areas, where the average number of sensible APs could easily reach 7.5 just as we show in Table II.

C. Evaluation of Cellular RSSI-Based Forgery Detection

Because of the similarity of cellular RSSI and WiFi RSSI data, we used the same experimental procedure as the WiFi scenario in this part of the experiment.

Datasets: Experimental data were collected in a village near Nanjing, China. A total of 10 LTE AP signals can be received near this village. We collected walking, cycling, and driving one-minute trajectories in this village at 2 s intervals. For each mode of transportation, we collect 5,000 trajectories. Then, as in the WiFi scenario, the training and test sets are divided for experiments.

We design experiments to discuss the influence of parameter r_2 and density of reference points on detection accuracy.

The Influence of Reference Radius r_2: Because the main idea of the algorithm is to use the information near the trajectory as a reference, we hope that the size of the reference area is reasonable so that valid historical points can be found in the reference area. If r_2 is too large, then the extraneous location points will have a bad influence on the judgment. If r_2 is too small, there will not be enough points to aid judgment.

We conduct experiments in the range of $r_2 \in [3.6, 6.6]$ in increments of $0.1\ m$ each time, and observe the change in accuracy. The result is shown in the Fig. 8(a). As r_2 increases, as more and more reference points are introduced into the judgment, the accuracy of trajectory detection gradually increases. When $r_2 > 5.8\,m$, the trajectory detection accuracy gradually decreases as more and more irrelevant historical points are introduced into the judgment. On the whole, when $r_2 = 5.6\,m$, the experiment achieves the maximum accuracy.

The Influence of Reference Points Density: As we show in Section IV, density of reference points is a significant challenge for detection. Here we perform detection experiments by modifying the average reference point density of the trajectory by randomly pruning some reference points. The experimental results are shown in the Fig. 8(b). The results show that the detection accuracy increases with the density in the three cases in suburban areas. When the density is greater than $0.07/m^2$, the detection accuracy of each region is greater than 85%, and this density condition can be satisfied for most areas.

Finally, we use WiFi RSSI based detection when the average number of WiFi APs around the trajectory is above 7.5 and the reference point density is larger than $0.2/m^2$. Otherwise, We use cellular RSSI-based detection. The results for three scenarios are presented in Fig. 9. As shown in the figure, the overall accuracy of the WiFi RSSI detection scheme is higher than that of the cellular RSSI detection scheme. This is because the historical signal point density of the dataset collected in the WiFi scenario is much higher than that of the Cellular dataset, and the number of WiFi APs that can receive signals at the same location is much larger than the number of Cellular APs. In addition, in the WiFi scenario, $Acc_{walking} > Acc_{cycling} > Acc_{driving}$. This is because the driving mode is farther away from the APs on both sides of the road, as shown in Fig. 7(b), the maximum historical signal point density gradually decreases. In the cellular scene, $Acc_{walking} < Acc_{cycling} < Acc_{driving}$. The average distance between trajectory points is larger in the driving mode, which makes it necessary for the attacker to forge RSSI. In the walking mode, the distance between the trajectory points is much smaller than the radiation range of the cellular AP, which makes the RSSI between most of the front and rear trajectory points almost unchanged, which brings greater detection difficulty.

Communication Overhead: We calculated and detected the communication overhead in a real-world scenario based on the

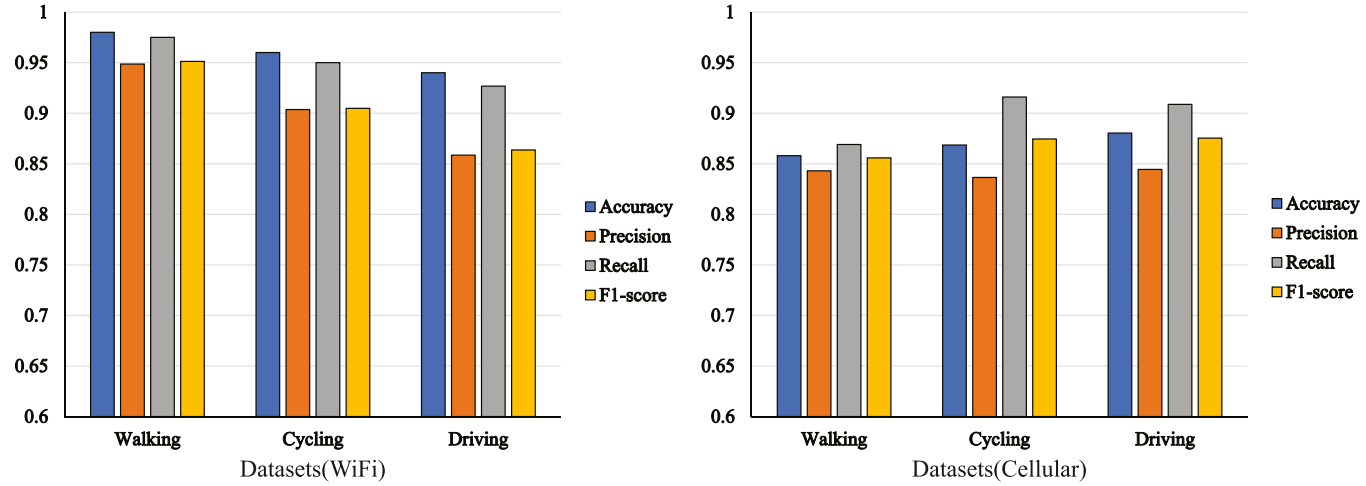


Fig. 9. Performance of our detection scheme.

above. The trajectories used in our experiments are one-minute trajectories where each trajectory contains 30 position points, and the sampling interval is 2 s. If a trajectory is longer than a minute, we will sample it several times, where the user is required to upload one minute of continuous trajectories each time. The size of a one-minute trajectory containing only position information is 2.25 KB. According to the conclusions of the above experiments, we choose $k = 10$ as the maximum number of WiFi APs that users need to upload at each location. In the worst case, where 10 WiFi RSSI and 4 cellular RSSI data are uploaded per location point, 13.0 KB of data will be uploaded per detection. Whether a WiFi network or a 3/4/5 G network, this communication overhead is much smaller than the regular uplink rate.

VI. Related Work

There have been some researches related to geographic location security that can be used to identify the authenticity of the trajectory.

Methods Based on Specific AP Equipment: He and Lin [11], Kanza [12] and others proposed a kind of methods to set up a special communication device as a verification device to ensure the authenticity of the location where the user needs to perform location verification. These devices often only support connections and communications within a certain distance, and prevent replay attacks based on a special protocol containing encrypted time stamp information. Pham et al. [4] proposed SecureRun, combined with the effective communication distance of the AP device, to form a continuous position proof on the user's activity track to ensure the authenticity of the position movement. This type of method requires a huge cost, and it is impossible for LBSP to deploy a large number of AP devices.

Methods Based on Communication Between Users: Talasila et al. [13] proposed a Bluetooth connection-based method to verify the user's location, allowing users who need to prove their location to establish a Bluetooth connection with users who use applications around to prove that they are indeed located in the claimed area. Xiao et al. [14] proposed to allow users in the same area to perform encrypted communication with each other to verify each other. One user selects certain signal sources and requires the other to provide the signal characteristics of these signal sources, and then calculates to determine whether the two parties are in the same area in real time. Wang et al. [15] proposed to introduce CA to put the process of comparing the environmental signal fields declared by both parties on the remote server to prevent the prover and the witness from deceiving in partnership. Most applications cannot meet the requirement that when a user initiates a request, other users who install the application are always nearby and the application happens to be running.

Methods Based on Environmental Signal: Zhang et al. [16] proposed that users upload RSSI information when check-in, and then use the historical information and current information of this location tag to perform density clustering. This method requires a location tag as the clustering center, and cannot defend against attacks that modify GPS coordinates. Zheng et al. [17] and Li et al. [18] proposed to generate credentials based on specific fields of real-time WiFi, cellular and other broadcast data packets, allowing users in the same area to verify each other's location. Brassil et al. [19], [20] proposed to verify the location by analyzing the flow data of wireless networks, base station signals and even sound waves. Abdou et al. [21], [22] proposed to calculate whether the positioning data of the device is reasonable based on the calculation of the communication delay time between the device and the base station, WiFi router and other AP devices, and realized the possible forgery detection strategy method [23].

Methods Based on Rules: He et al. [36] proposed a method of heuristic rules for trajectories and requests to distinguish whether users are cheating. Polakis et al. [37] also proposed some similar rule-based detection schemes. These rules include whether the movement speed is too fast, whether the active request is too frequent, and so on. This kind of method has simple logic and low cost, but it is vulnerable to replay attacks. The attacker only needs to record the GPS sequence of a historical

trajectory of his real movement, and replay it in sequence at the corresponding time interval.

In addition, Pelechrinis et al. [38] proposed to set up some HoneyPot for FourSquare that does not exist in the storefront to induce malicious users to fake location attacks. [39] requires users to connect to the WiFi of the FourSquare store and scan the QR code to get another coupon. [40], [41], [42], [43], [44], [45], [46], [47] respectively proposed some indoor positioning methods. Because the trajectory basically occurs outdoors, a variety of influencing factors must be considered when using WiFi outdoors, and the WiFi strength attenuation will also be irregular due to different terrains. So these methods are not applicable.

VII. Conclusion

This work introduces the security risks and defenses of GPS trajectories. First of all, we use adversarial examples attacks on the GPS trajectories from the perspective of the attacker, which proves that the trajectory detection cannot be accurately performed using only GPS motion characteristics in the current network environment. Then we propose a defense scheme against adversarial example replay attacks, and we introduce radio RSSIs as proof of trajectories. Through theoretical analysis and experimental evaluation, we prove that this solution has good defensive performance in the face of GPS trajectory forgery.

Acknowledgments

This work is an extended version of [1] (ICDCS 2022).

References

[1] H. Yang, Z. Xia, J. Shin, J. Hua, Y. Mao, and S. Zhong, "Are you moving as you claim: GPS trajectory forgery and detection in location-based services," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst.*, 2022, pp. 1166–1176.

[2] H. Yu, H. Zhang, X. Jia, X. Chen, and X. Yu, "pSafety: Privacy-preserving safety monitoring in online ride hailing services," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 209–224, Jan./Feb. 2023.

[3] J. N. Gilmore, "Securing the kids: Geofencing and child wearables," *Convergence*, vol. 26, no. 3, 2019, Art. no. 135485651988231.

[4] A. Pham, K. Huguenin, I. Bilogrevic, I. Dacosta, and J.-P. Hubaux, "SecureRun: Cheat-proof and private summaries for location-based activities," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 2109–2123, Aug. 2016.

[5] Z. Chen, B. Wei, and J. Quan, "A travel assistant application based on Android Baidu map," in *Proc. IEEE Int. Conf. Intell. Comput. Automat. Syst.*, 2020, pp. 299–303.

[6] H. Huang et al., "Dynamic path planning based on improved D* algorithms of Gaode map," in *Proc. IEEE 3rd Inf. Technol. Netw. Electron. Automat. Control Conf.*, 2019, pp. 1121–1124.

[7] C. Y. Adegoke, "Uber drivers in Lagos are using a fake GPS app to inflate rider fares," 2017. Accessed: Nov. 14, 2017. [Online]. Available: https://qz.com/africa/1127853/uber-drivers-in-lagos-nigeria-use-fake-lockito-app-to-boost-fares/

[8] C. Ozkan and K. Bicakci, "Security analysis of mobile authenticator applications," in *Proc. IEEE Int. Conf. Inf. Secur. Cryptol.*, 2020, pp. 18–30.

[9] G. M. Zhou, M. Duan, Q. Xi, and H. Wu, "ChanDet: Detection model for potential channel of iOS applications," *J. Phys.: Conf. Ser.*, vol. 1187, 2019, Art. no. 042045.

[10] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proc. 10th Workshop Mobile Comput. Syst. Appl.*, New York, NY, USA, 2009, Art. no. 3.

[11] X. Lin and W. He, "WiLoVe: A WiFi-coverage based location verification system in LBS," *Procedia Comput. Sci.*, vol. 34, pp. 484–491, 2014.

[12] Y. Kanza, "Location corroborations by mobile devices without traces," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, New York, NY, USA, 2016, Art. no. 60.

[13] M. Talasila, R. Curtmola, and C. Borcea, "Collaborative bluetooth-based location authentication on smart phones," *Pervasive Mobile Comput.*, vol. 17, pp. 43–62, 2015.

[14] L. Xiao, Q. Yan, W. Lou, G. Chen, and Y. T. Hou, "Proximity-based security techniques for mobile users in wireless networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 12, pp. 2089–2100, Dec. 2013.

[15] X. Wang, A. Pande, J. Zhu, and P. Mohapatra, "STAMP: Enabling privacy-preserving location proofs for mobile users," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3276–3289, Dec. 2016.

[16] K. Zhang, W. Jeng, F. Fofie, K. Pelechrinis, and P. Krishnamurthy, "Towards reliable spatial information in LBSNs," in *Proc. ACM Conf. Ubiquitous Comput.*, New York, NY, USA, 2012, pp. 950–955.

[17] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "Location based handshake and private proximity test with location tags," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 4, pp. 406–419, Jul./Aug. 2017.

[18] Y. Li, L. Zhou, H. Zhu, and L. Sun, "Privacy-preserving location proof for securing large-scale database-driven cognitive radio networks," *IEEE Internet Things J.*, vol. 3, no. 4, pp. 563–571, Aug. 2016.

[19] J. Brassil, R. Netravali, S. Haber, P. Manadhata, and P. Rao, "Authenticating a mobile device's location using voice signatures," in *Proc. IEEE 8th Int. Conf. Wirel. Mobile Comput. Netw. Commun.*, 2012, pp. 458–465.

[20] J. Brassil, P. K. Manadhata, and R. Netravali, "Traffic signature-based mobile device location authentication," *IEEE Trans. Mobile Comput.*, vol. 13, no. 9, pp. 2156–2169, Sep. 2014.

[21] A. M. Abdou, A. Matrawy, and P. C. van Oorschot, "Location verification on the internet: Towards enforcing location-aware access policies over internet clients," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2014, pp. 175–183.

[22] A. Abdou, A. Matrawy, and P. C. van Oorschot, "CPV: Delay-based location verification for the internet," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 2, pp. 130–144, Mar./Apr. 2017.

[23] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Accurate manipulation of delay-based internet geolocation," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, New York, NY, USA, 2017, pp. 887–898.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[25] C. Szegedy et al., "Intriguing properties of neural networks," 2014, *arXiv:1312.6199*.

[26] J. E. Vargas-Munoz, S. Srivastava, D. Tuia, and A. X. Falcão, "OpenStreetMap: Challenges and opportunities in machine learning and remote sensing," *IEEE Geosci. Remote Sens. Mag.*, vol. 9, no. 1, pp. 184–199, Mar. 2021.

[27] L. R. Medsker and L. Jain, "Recurrent neural networks," *Des. Appl.*, vol. 5, pp. 64–67, 2001.

[28] V. Palazón and A. Marzal, "Speeding up shape classification by means of a cyclic dynamic time warping lower bound," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.*, E. Corchado, H. Yin, V. Botti, and C. Fyfe, Eds., Berlin, Germany: Springer, 2006, pp. 436–443.

[29] Z. Zhang, K. Huang, and T. Tan, "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *Proc. IEEE 18th Int. Conf. Pattern Recognit.*, 2006, pp. 1135–1138.

[30] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.

[31] M. Y. Karim, H. Kagdi, and M. Di Penta, "Mining Android apps to recommend permissions," in *Proc. IEEE 23rd Int. Conf. Softw. Anal. Evol. Reengineering*, 2016, pp. 427–437.

[32] M. Lutaaya, "Rethinking app permissions on iOS," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, New York, NY, USA, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3170427.3180284

[33] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, New York, NY, USA, 2016, pp. 785–794.

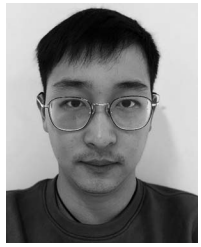
[34] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Sys6GJqxl

[35] Q. Zhao et al., "Synthesizing ReLU neural networks with two hidden layers as barrier certificates for hybrid systems," in *Proc. 24th Int. Conf. Hybrid Syst.: Computation Control*, 2021, pp. 1–11.

[36] W. He, X. Liu, and M. Ren, "Location cheating: A security challenge to location-based social network services," in *Proc. IEEE 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 740–749.
[37] I. Polakis, S. Volanis, E. Athanasopoulos, and E. P. Markatos, "The man who was there: Validating check-ins in location-based services," in *Proc. 29th Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, 2013, pp. 19–28.
[38] K. Pelechrinis, P. Krishnamurthy, and K. Zhang, "Gaming the game: Honeypot venues against cheaters in location-based social networks," *CoRR*, vol. abs/1210.4517, 2012. [Online]. Available: http://arxiv.org/abs/1210.4517
[39] B. Carbunar and R. Potharaju, "You unlocked the Mt. Everest badge on foursquare! Countering location fraud in geosocial networks," in *Proc. IEEE 9th Int. Conf. Mobile Ad-Hoc Sensor Syst.*, 2012, pp. 182–190.
[40] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proc. 3rd Int. Conf. Mobile Syst. Appl. Serv.*, New York, NY, USA, 2005, pp. 205–218.
[41] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2012, pp. 293–304.
[42] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: Wireless indoor localization with little human intervention," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2012, pp. 269–280.
[43] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Serv.*, New York, NY, USA, 2012, pp. 197–210.
[44] C. Wu, Z. Yang, and Y. Liu, "Smartphones based crowdsourcing for indoor localization," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 444–457, Feb. 2015.
[45] J. Niu, B. Wang, L. Cheng, and J. J. P. C. Rodrigues, "WicLoc: An indoor localization system based on WiFi fingerprints and crowdsourcing," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 3008–3013.
[46] L. Li, X. Guo, N. Ansari, and H. Li, "A hybrid fingerprint quality evaluation model for WiFi localization," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9829–9840, Dec. 2019.
[47] M. B. Kjaergaard and C. V. Munk, "Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength (concise contribution)," in *Proc. IEEE 6th Annu. Int. Conf. Pervasive Comput. Commun.*, 2008, pp. 110–116.

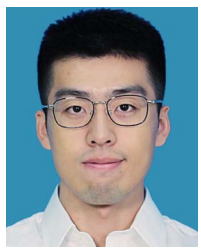
Huaming Yang (Student Member, IEEE) received the BS degree in computer science from Nanjing University, in 2018. He is currently working toward the PhD degree with the Department of Computer Science and Technology, Nanjing University. His current research interests include network security, location security, and mobile security.

Zhongzhou Xia (Student Member, IEEE) received the BE degree in software engineering from Sun Yat-sen University, in 2018, and the MS degree in computer technology from Nanjing University, in 2021. His current research interests include network security, cryptography, and operating system.



Jersy Shin (Student Member, IEEE) received the BS degree in computer science from Nanjing University, in 2018. His current research interests include machine learning security and differential privacy.

Jingyu Hua (Member, IEEE) received the BE and ME degrees in software engineering from the Dalian University of Technology, China, in 2007 and 2009, respectively, and the PhD degree in informatics from Kyushu University, Japan, in 2012. His current research interests include security and privacy in mobile computing, and system security.



Yunlong Mao (Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, Nanjing, China, in 2013 and 2018, respectively. He is currently an assistant researcher with the Department of Computer Science and Technology, Nanjing University. His current research interests include security, privacy, and machine learning.

Sheng Zhong (Senior Member, IEEE) received the BS and MS degrees in computer science from Nanjing University, in 1996 and 1999, respectively, and the PhD degree in computer science from Yale University, in 2004. His research interests include security, privacy, and economic incentives.