2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS) | 978-1-6654-7177-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDCS54860.2022.00115

Are You Moving as You Claim: GPS Trajectory Forgery and Detection in Location-Based Services

Huaming Yang†, Zhongzhou Xia†, Jersy Shin†, Jingyu Hua†*, Yunlong Mao†*, and Sheng Zhong†
†State Key Laboratory for Novel Software Technology, Nanjing University, China
yhmnjucs@163.com, losingle@qq.com, jersyshin@foxmail.com, {huajingyu, maoyl, zhongsheng}@nju.edu.cn

***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 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 a reasonable walking, cycling, or driving route when being projected to the map. We show through experiments that they can hardly be detected by mainstream trajectory service providers, even after being equipped with machine learning-based approaches. Therefore, we further present a dedicated countermeasure by validating the reasonability of reported received signal strength indicator (RSSI) data of WiFi access points (APs) nearby every location. It can well deal with the most challenging replay scenario, which can hardly be handled by existing WiFi-based location verification methods. We conduct extensive real-world experiments in three local commercial areas covering walking, cycling, and driving scenarios. Results demonstrate the high detection accuracy of this method.**

***Index Terms*—Trajectory adversarial examples; Trajectory forgery attacks; Trajectory detection**

I. Introduction

A lot of mobile apps require to 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 [1]. 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 [2]. Fitness apps such as Keep rely on users' trajectories to calculate and show their jogging statistics (miles, velocities, etc.) and even launch competitions among friends [3]. Actually, due to the high value of trajectories, many leading location service providers (LSP, e.g., Baidu [4] and Amap [5] in China) construct open trajectory platforms, which implement basic capabilities of trajectory collection, storage, and analysis. App developers can leverage SDKs and cloud APIs to quickly build their own trajectory-based services and systems.

This work was supported in part by NSFC-61972195, the Leading-edge Technology Program of Jiangsu NSF (No.BK20202001), NSFC-61872179, and NSFC-61872176.

*Jingyu Hua and Yunlong Mao are corresponding authors.

As the trajectories are uploaded by clients, a potential risk that service providers have to consider is whether the trajectories received are forged by users. Malicious users have both the motivation and the ability to launch such trajectory spoofing attacks in many apps. First, such attacks can bring them 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 [6]. Second, as the app is running on the devices fully controlled by users, it is not difficult for them to launch trajectory spoofing attacks from various ways, e.g., using hooking frameworks (Frida [7], Substrate [8], 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 given manually.

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. We can extend them to detect forgery trajectories by independently checking each 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 [3], [9]–[11] 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), and it is too expensive to deploy so many APs that can cover a large enough area. Rather than dedicated equipment, the second category [12]–[14] requires users to certify their locations by performing mutual peer-to-peer communication with another user within a certain distance.

2575-8411/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDCS54860.2022.00115

Nevertheless, it is hard to guarantee that there are always peers appearing nearby. The last category [15]–[22] 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 easily escape from being detected by replaying their historical data with slight noises. Besides above, trajectories expose extra information such as the motion characteristics (velocity, direction, etc.) of users, which can also be used for forgery detection. In general, none of these existing solutions can meet the needs of trajectory detection in current application scenarios.

Targeting this problem, this paper takes the perspective of attackers to consider how they will fabricate fake trajectories rather than single fake points. Only if we know clearly about 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, the LSP could extract motion features from a given trajectory, which means the attacker has to guarantee such features 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 first 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 that the provider may use, the attacker can also train a classification model (a LSTM-based classifier in our approach [23]) using various public trajectory datasets. Then, a natural approach to meeting the first requirement is to utilize the adversarial examples technique [24] 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 can collect the RSSIs of nearby APs at each location 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 [15]. 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 2.5m radius is above 4, which we think is reasonable in most commercial areas.

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. In other words, the provider can only rely on the data uploaded by clients to decide whether a trajectory is real or not. We will not consider the arms race on the client-side.

Our assumption is reasonable as the app is running on devices fully controlled by users. It is not difficult for a user to use various hooking frameworks (Frida, Substrate, 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 malicious users may have gained the root privilege. In extreme cases, they may even install external analog GPS equipment, in which the positioning results are fully defined by manual. As a result, a really effective defense method should be on the server-side.

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. These datasets can be used to learn motion characteristics of humans, and thus are extremely important for both trajectory forgery and detection.

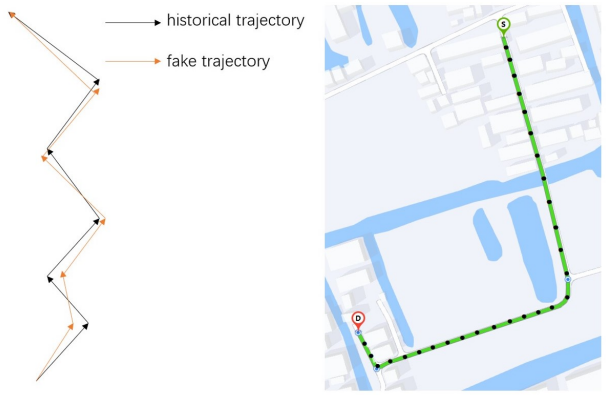
Attack goal. Given a source S, a destination D and a time sequence $[t_1, t_2, ..., t_n]$ where the time interval is a constant c: $c = t_{i+1} - t_i$, our task aims to generate a fake trajectory T=$[P_1, P_2, ..., P_n]$, where P_i is the geo-coordinate of the user at time t_i , and $P_1 = S$,$P_n = D$, 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:

Motion characteristics indistinguishability. As lots 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) 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.

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. 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 predetermined 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. Because Dynamic Time Warping (DTW [25]) is the most often used metric to measure the similarity between two temporal sequences, we also employ it 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:

$$loss(T, T') = \lambda_1 * loss_{ent}(T, T') + DTW(T, T'), \quad (1)$$

where $loss_{ent}(T, T')$ is the cross-entropy loss function which describes the loss of classification errors, λ_1 is a constant parameter.

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.

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

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:

$$loss(T, T') = \lambda_2 * loss_{ent}(T, T') + loss_2(T, T'), \quad (3)$$

where λ_2 is a constant parameter.

Adversarial trajectory generation. We use the optimization based C&W algorithm [26] 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.

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 [27], [28]. 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 Sec. IV-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=$[P_1, P_2, \cdots, P_n]$ 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, $RSSI_i$=$[rssi_1$,$rssi_2, \cdots, rssi_m]$ and MAC_i=$[mac_1, mac_2, \cdots, mac_m]$ 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 $\mathbb{H} = \{H_1, H_2, \cdots, H_k\}$ 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 $O \in T$, our scheme tries to leverage the RSSIs of those historical points within a circle of radius r around O to estimate the confidences of its reported RSSI values. We call such points *reference points* and the circle $C_O(r)$ *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 O 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.

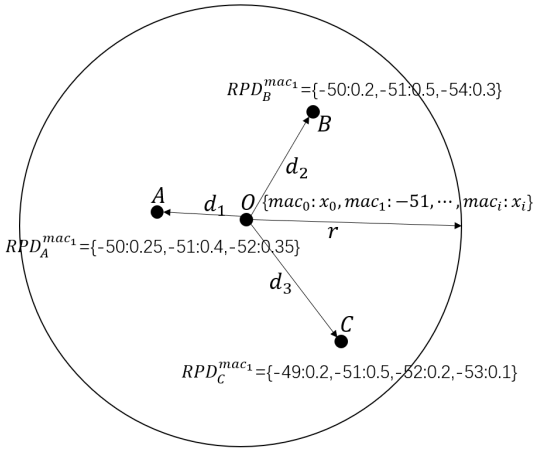


Fig. 2. RSSI verification based on adjacent historical reference points

First, due to GPS errors, we actually do not know the exact positions of both O 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

$$RPD_H^{mac_i}(x) = \frac{\|\{Q \in \mathbb{H} | Q.rssi_i = x \wedge Q \in C_H(R)\}\|}{\|C_H(R)\|}, \tag{4}$$

which is just identical to the ratio of historical points having the specific RSSI value x within this area. We call $C_H(R)$ the RPD counting area. With this distribution, the estimated confidence of $O.rssi_i$, the reported RSSI of AP mac_i at position O, according to the reference point H is $RPD_H^{mac_i}(O.rssi_i)$.

Because there might be more than one reference point in $C_O(r)$, 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 $\theta_1(H,O)$ to consider this fact in the final RSSI confidence calculation: given a reference point $H \in \mathbb{H}$,

$$\theta_1(H,O) = \frac{\frac{1}{d_{euc}(H,O)}}{\sum_{H \in C_O(r)} \frac{1}{d_{euc}(H,O)}}, \tag{5}$$

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

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:

$$\theta_2(H) = 1 - (\frac{1}{t})^{\epsilon}, \tag{6}$$

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 O:

$$\Phi_O(O.rssi_i) = \sum_{H \in C_O(r) \wedge \mathbb{H}} \theta_1(H,O) * \theta_2(H) * RPD_H^{mac_i}(O.rssi_i). \tag{7}$$

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

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

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

So, the final feature vector for the whole trajectory is

$$feature = [feat_1, feat_2, ..., feat_n]. \tag{8}$$

We finally train an XGBoost-based [29] binary trajectory classification model with this feature vector to detect fake trajectories. For this purpose, as we mentioned in Sec. IV, 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 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 over 500 GPS coordinates at the same position. We take the average coordinate as the real position, according to the central limit theorem, the distance d between other GPS coordinates and the average coordinate obeys unilateral normal distribution $d \sim N(0,\sigma^2)$. On the basis of the three-sigma rule, 99.7% of the coordinates are within $d < 3\sigma$. Therefore, we define the maximum position deviation *R* = 6σ = 3m.

IV. 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. Openstreetmap is a free and open-source platform jointly created by the Internet public [30]. We downloaded the trajectories within two months from June 2020 to August 2020 as the real dataset. Because the real trajectory is irregular, we use 1s 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 [5]. 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 1s 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: Naturally, we employ LSTM to generate our target classifier. LSTM is a special kind of RNN [31] that can learn long-term rules and process sequential data (temperature, traffic volume, sales, etc.). LSTM is usually used for vehicle driving prediction, natural language processing, text matching, and other problems.

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 Sec. 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.

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=$[P_1, P_2, ..., P_n]$, 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 Table I, which are extremely accurate.

3) Evaluation of the proposed forgery method: When we apply C&W [26] 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 Equation 1 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.3.

TABLE I
CLASSIFICATION PERFORMANCE AGAINST NAIVE ATTACKS

Classifiers	Accuracy	Precision	Recall	F1-score
C	0.9886	0.9982	0.979	0.9885
XGBoost	0.9542	0.9693	0.9538	0.9615
LSTM-1	0.9874	0.9885	0.9905	0.9895
LSTM-2	0.9909	0.9922	0.9926	0.9924

Model C is the target model of trajectory adversarial examples attacks.

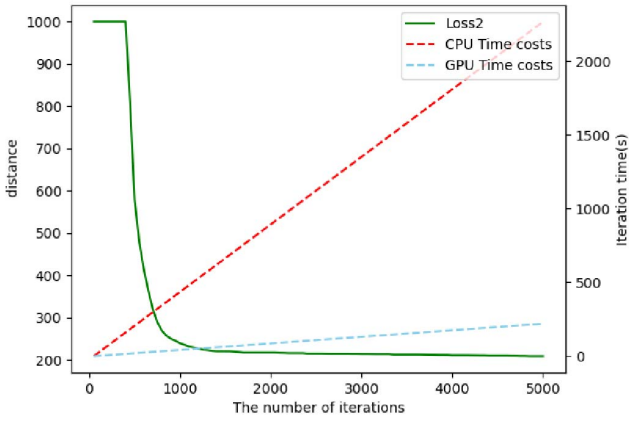


Fig. 3. Variation curve with the number of iterations

When the number of iterations ≤ 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.

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 200m 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 formula 1 and λ_2 in formula 3) 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 II).

4) Transferability evaluation: We further train other completely unrelated detection models to verify the transferability of the attack [33].

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

TABLE II
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%

TABLE III
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

- **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 Sec. IV-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 Table I.

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 Table I.

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 [32]. The performance of LSTM-2 against naive attacks is shown in Table I.

Finally, we select 1,000 real trajectories from D_{train} for replay attacks and 1,000 fake trajectories from AN for navigation attacks (Sec. IV-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 II. The results show that our forgery scheme has good transferability.

Experiments prove that our attack is effective. Trajectory adversarial examples 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-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 2s. 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 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 III.

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 Sec. 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 Sec. 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) 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. 4. When $r < 1m$, 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 $r > 1m$, due to the expansion of the reference area, the number of points available for reference

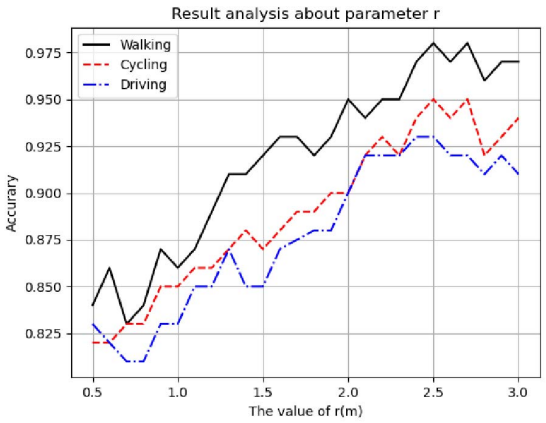


Fig. 4. Influence of r

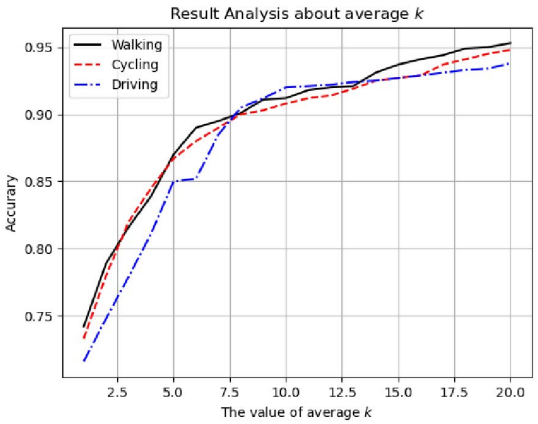


Fig. 6. Influence of average k: the AP density

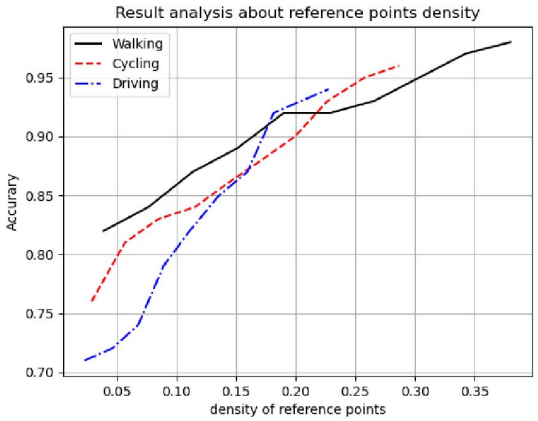


Fig. 5. Influence of the reference points density

is also increasing, the accuracy rate continues to rise and the maximum value is obtained when $r = 2.5m$. When $r > 2.5m$, 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. 5

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

Finally, we set $r = 2.5$m. The results for three scenarios are presented in Table IV.

TABLE IV
PERFORMANCE OF OUR DETECTION SCHEME

	Accuracy	Precision	Recall	F1-score
Walking	0.98	0.9286	0.975	0.9512
Cycling	0.96	0.8636	0.95	0.9048
Driving	0.94	0.8085	0.9268	0.8636

V. 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 [10], Kanza [11] 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.* [3] 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.* [12] 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.* [13] 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.* [14] 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.* [15] 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.* [16] and Li *et al.* [17] 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.* [18], [19] proposed to verify the location by analyzing the flow data of wireless networks, base station signals and even sound waves. Abdou *et al.* [20], [21] 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 [22].

Methods based on rules. He *et al.* [34] proposed a method of heuristic rules for trajectories and requests to distinguish whether users are cheating. Polakis *et al.* [35] 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.* [36] proposed to set up some HoneyPot for FourSquare that does not exist in the storefront to induce malicious users to fake location attacks. [37] requires users to connect to the WiFi of the FourSquare store and scan the QR code to get another coupon. [38]–[45] 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.

VI. Conclusions

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 WiFi 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.

References

[1] H. Yu, H. Zhang, X. Jia, X. Chen, and X. Yu, "psafety: Privacy-preserving safety monitoring in online ride hailing services," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021.

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

[3] A. Pham, K. Huguenin, I. Bilogrevic, I. Dacosta, and J.-P. Hubaux, "Securerun: Cheat-proof and private summaries for location-based activities," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2109–2123, 2016.

[4] Z. Chen, B. Wei, and J. Quan, "A travel assistant application based on android baidu map," in *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, 2020, pp. 299–303.

[5] H. Huang, P. Huang, S. Zhong, T. Long, S. Wang, E. Qiang, Y. Zhong, and L. He, "Dynamic path planning based on improved d* algorithms of gaode map," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 1121–1124.

[6] C. Yemisi Adegoke, "Uber drivers in lagos are using a fake gps app to inflate rider fares," 2017, accessed November 14, 2017. https://qz.com/africa/1127853/uber-drivers-in-lagos-nigeria-use-fake-lockito-app-to-boost-fares/.

[7] C. Ozkan and K. Bicakci, "Security analysis of mobile authenticator applications," in *2020 International Conference on Information Security and Cryptology (ISCTURKEY)*, 2020, pp. 18–30.

[8] G. M. Zhou, M. Duan, Q. Xi, and H. Wu, "Chandet: Detection model for potential channel of ios applications," *Journal of Physics: Conference Series*, vol. 1187, pp. 042 045–, 2019.

[9] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '09. New York, NY, USA: Association for Computing Machinery, 2009.

[10] X. Lin and W. He, "Wilove: A wifi-coverage based location verification system in lbs," *Procedia Computer Science*, vol. 34, pp. 484–491, 2014, the 9th International Conference on Future Networks and Communications (FNC'14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops.

[11] Y. Kanza, "Location corroborations by mobile devices without traces," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPACIAL '16. New York, NY, USA: Association for Computing Machinery, 2016.

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

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

[14] X. Wang, A. Pande, J. Zhu, and P. Mohapatra, "Stamp: Enabling privacy-preserving location proofs for mobile users," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3276–3289, 2016.

[15] K. Zhang, W. Jeng, F. Fofie, K. Pelechrinis, and P. Krishnamurthy, "Towards reliable spatial information in lbsns," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 950–955.

[16] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "Location based handshake and private proximity test with location tags," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 4, pp. 406–419, 2017.

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

[18] J. Brassil, R. Netravali, S. Haber, P. Manadhata, and P. Rao, "Authenticating a mobile device's location using voice signatures," in *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2012, pp. 458–465.

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

[20] 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 *2014 IEEE Conference on Communications and Network Security*, 2014, pp. 175–183.

[21] A. Abdou, A. Matrawy, and P. C. van Oorschot, "Cpv: Delay-based location verification for the internet," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 2, pp. 130–144, 2017.

[22] ——, "Accurate manipulation of delay-based internet geolocation," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 887–898.

[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, and R. Fergus, "Intriguing properties of neural networks," 2013.

[25] V. Palazón and A. Marzal, "Speeding up shape classification by means of a cyclic dynamic time warping lower bound," in *Intelligent Data Engineering and Automated Learning – IDEAL 2006*, E. Corchado, H. Yin, V. Botti, and C. Fyfe, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 436–443.

[26] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[27] M. Y. Karim, H. Kagdi, and M. Di Penta, "Mining android apps to recommend permissions," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1, 2016, pp. 427–437.

[28] M. Lutaaya, "Rethinking app permissions on ios," in *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1?6. [Online]. Available: https://doi.org/10.1145/3170427.3180284

[29] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794.

[30] J. E. Vargas-Munoz, S. Srivastava, D. Tuia, and A. X. Falcão, "Openstreetmap: Challenges and opportunities in machine learning and remote sensing," *IEEE Geoscience and Remote Sensing Magazine*, vol. 9, no. 1, pp. 184–199, 2021.

[31] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[32] Q. Zhao, X. Chen, Y. Zhang, M. Sha, Z. Yang, W. Lin, E. Tang, Q. Chen, and X. Li, "Synthesizing relu neural networks with two hidden layers as barrier certificates for hybrid systems," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.

[33] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," 2016.

[34] W. He, X. Liu, and M. Ren, "Location cheating: A security challenge to location-based social network services," in *2011 31st International Conference on Distributed Computing Systems*, 2011, pp. 740–749.

[35] I. Polakis, S. Volanis, E. Athanasopoulos, and E. P. Markatos, "The man who was there: Validating check-ins in location-based services," in *Proceedings of the 29th Annual Computer Security Applications Conference*, ser. ACSAC '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 19–28.

[36] K. Pelechrinis, P. Krishnamurthy, and K. Zhang, "Gaming the game: Honeypot venues against cheaters in location-based social networks," 2012.

[37] B. Carbunar and R. Potharaju, "You unlocked the mt. everest badge on foursquare! countering location fraud in geosocial networks," in *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, 2012, pp. 182–190.

[38] M. Youssef and A. Agrawala, "The horus wlan location determination system," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. New York, NY, USA: Association for Computing Machinery, 2005, pp. 205–218.

[39] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 293–304.

[40] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: Wireless indoor localization with little human intervention," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 269–280.

[41] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 197–210.

[42] C. Wu, Z. Yang, and Y. Liu, "Smartphones based crowdsourcing for indoor localization," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 444–457, 2015.

[43] J. Niu, B. Wang, L. Cheng, and J. J. P. C. Rodrigues, "Wicloc: An indoor localization system based on wifi fingerprints and crowdsourcing," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 3008–3013.

[44] L. Li, X. Guo, N. Ansari, and H. Li, "A hybrid fingerprint quality evaluation model for wifi localization," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9829–9840, 2019.

[45] M. B. Kjaergaard and C. V. Munk, "Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength (concise contribution)," in *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2008, pp. 110–116.