LightSense: A Novel Side Channel for Zero-permission Mobile User Tracking

Quanqi Ye[1], Yan Zhang[2], Guangdong Bai[3], Naipeng Dong[4], Zhenkai Liang[4], Jin Song Dong[4], and Haoyu Wang[5]

[1] Advanced Digital Sciences Centre, Singapore
quanqi.ye@adsc-create.edu.sg
[2] University of Science and Technology of China
zhangyan.aq@gmail.com
[3] The University of Queensland, Australia
g.bai@uq.ed.au
[4] National University of Singapore
{dcsdn,dcsdjs}@nus.edu.sg
liangzk@comp.nus.edu.sg
[5] Beijing University of Posts and Telecommunications
haoyuwang@bupt.edu.cn

Abstract. Android devices are equipped with various sensors. Permissions from users must be explicitly granted for apps to obtain sensitive information, e.g., geographic location. However, some of the sensors are considered trivial such that no permission control is enforced over them, e.g., the ambient light sensor. In this work, we present a novel side channel, i.e. the ambient light sensor, that can be used to track the mobile users. We develop a location tracking system with off-line trained route identification models using the values from the attacker's own ambient light sensor. The system can then be used to track a user's geographic location. The experiment results show that our route identification models achieve a high accuracy of over 91% in user's route identification and our tracking system achieves an accuracy at about 64% in real-time tracking the user with estimation error at about 70 meters. Our system out-performs the state-of-the-art works with other side channels. Our work shows that with merely the values from the ambient light sensor of user's mobile phone that requires zero-permission to access, the geographic routes that the user has taken and his/her real-time location can be identified with machine learning techniques in high accuracy.

1 Introduction

Mobile phones nowadays are equipped with many powerful sensors. The mobile sensing techniques bring in great convenience for app developers to enrich the functionalities of mobile apps, for example, the Location-Based-Services (LBSs). However, it also introduces security and privacy issues. The data collected by such apps might appear to be innocuous, but malicious apps could use it for other purposes. Recent reports show that the location information the malicious apps

collect may reveal the user's occupation or personal activities which poses severe privacy infringement to the users [15,6]. On the other hand, multiple incidents of massive personal private data leakage happen from time to time [31,17,10]. Therefore, mobile users are becoming increasingly concerned about what data the mobile apps can collect and whether it can infringe their privacies.

Android operating system (Android for short) uses a permission model to regulate apps' access to sensitive data. For example, if an app needs to access a user's geographical locations, it needs to explicitly declare the *ACCESS_FINE_LOCATION* permission in its manifest file. When a user installs the app on his/her mobile phone, a request to the permission will be shown to the user for approval. The user can choose to accept the permission request and installs the app or he/she can simply decline the request. Android 6.0 (API level 23) and above enforce runtime permission mechanism [3]. User can choose to approve or deny the access request to sensitive data during runtime when the app actually needs to access the sensitive data. In this way, a user can control the permissions granted to the apps in a more flexible way.

However, although most privacy-sensitive resources, e.g., geographical location, have been guarded by permissions, there are still some resources that are considered as innocuous and available to all apps without being guarded by any permission. The ambient light information is one of such resources. It can be detected by the ambient light sensor. The value of the ambient light sensor is a double precision number representing the ambient brightness level. Reading the value of ambient light sensor from the app does not need any permission in all versions of Android. Furthermore, almost all newly manufactured mobile phones come with this sensor to provide the feature of automatically adjusting the screen brightness level according to ambient brightness level. For example, when the user is standing under the sun, the screen brightness will be automatically increased so that the content displayed on screen becomes more readable.

In this work, we present a novel side channel (i.e., ambient light sensor) and we show that even the user denies the app from accessing the location information, the malicious apps on the mobile phone can still infer the location information of the mobile user from this side channel. This novel side channel can be used in the attacks to track the mobile users' locations in urban area.

The victims can be those who are walking texting, runners or hikers putting their mobile phones in the armband, and bikers who rest the mobile phones on phone brackets etc. It can also be applied in the scenarios where the mobile phone is not covered (e.g., resting on bracket for navigation). Furthermore, it is already an old news that the number of people texting while walking who are causing many traffic accidents is so large [23] that many cities in the world (e.g., Washington D.C.) have built "phone lanes" dedicated to people that are text walking [19]. Potentially, there are many victims affected by this side channel.

Taking advantages of machine learning techniques, the seemingly innocuous values collected from the ambient light sensor can be used to track the mobile user of the routes that he/she goes through and even locate the user along the

routes in real-time. Specifically, we prove that using this novel side channel — the ambient light sensor, an attacker is able to achieve the following two goals.

1. **Route Identification.** Given the known set of routes the victim[6] might go through, the attacker can identify which routes the victim has gone through with the ambient light sensor values from the victim's mobile phone.
2. **Real-time Location Tracking.** Given the known area where the victim might be travelling on, the attacker can infer the location of the victim in a real-time manner with the ambient light sensor values from the victim's mobile phone.

In order to achieve the above two goals, we first collect the values from the ambient light sensor on the experimental routes without interacting with victims. Then, we train models using machine learning algorithms off-line with the collected data. The trained models are later used to identify the routes that the user has gone through. After that, based on the route identification approach, we develop a location tracking system that can be used to track the user in real-time. We highlight that the above approach relies solely on the ambient light sensor whose values are available to all apps in Android without any permission. On one hand, this guarantees the stealthiness of the attack if an attacker applied the above approach. On the other hand, although combined sensor can provide more information and achieve better accuracy, by using a single sensor, we can evaluate "how much" information can be leaked just from the light sensor alone. Thus, we have a baseline on how severe is the privacy leakage from this side channel. This attack is also flexible as it can be launched as a targeted attack or a large scale attack (e.g., phishing by location-based advertisement services [14]).

Our evaluation shows that this approach can achieve a high overall accuracy of about 91% in route identification and accuracy of about 64% in real-time tracking the victim with estimation error at about 70 meters, which outperforms other existing the state-of-the-art works with other side channels (See Section 6). We summarize our contributions in the following.

1.1 Our Contributions

1. We discover and prove a novel side-channel that can be used to track mobile users. To the best of our knowledge, we are the first to show that the ambient light sensors on Android mobile devices can be used to track the mobile users.
2. We develop a location tracking system using machine learning techniques to track the mobile users.
3. The results in the real-world experiments suggest that our system achieves a much higher accuracy with less estimation error than previous similar works (section 6) with other side channels.

2 Threat Models and Challenges

We assume that the attacker has managed to have his malicious app installed on the victim's mobile device. This is practical and easy to achieve as the attacker

[6] We use "victim" and the aforementioned "user" interchangeably in this paper.

can develop a malicious app and disguise it as an useful utility app that does not require any special permission (e.g., a stock price watching app). He can then wait for the victims to install his app. The malicious app can monitor victim's light sensor in the background and continuously collects the values from the ambient light sensor. It does not require any special permission that needs to be granted by the user. The collected data can be stored in its private storage. Starting from Android 4.4, apps do not need any permission to store data in an app's own private storage [4]. The user may not even notice that he/she is being tracked. However, the malicious app does need the Internet connection to transmit the collected data to the attacker. To achieve this, the malicious app can request the Internet permission. For Android operating system with API level higher than 23, the Internet permission request is granted automatically if the user chooses to install an app which requests the Internet permission and it is hidden from users by default [2]. The Internet permission is classified under normal permissions category and normal permissions can not be revoked by users. Although the Internet permission is still listed on the permissions requested in the system setting menu after the app has been installed, users seldom care about what permissions an app has requested and seldom check the permission list. Even though the users may check the permission list, they might not consider the Internet permission as a suspicious permission request, given that the Internet permission is such a common permission request that 83% of apps request full network access permission [25].

The attacker can even remove the Internet permission request to make the app really zero-permission. This can be achieved by delegating the network communication task to the default browser in the system. When the app has data to upload to the attacker's server, it can generate an `intent` with an URL and attaches the data in the query string for system browser [20]. Then the app prompts a dialog for the user to update the app. Upon clicking the buttons, browser is launched, the url request with generated URL with the data embedded is passed to the attacker's server. After the server has received the data, it redirects the browser to the app's homepage. In this way, the user would not even notice that the data has been smuggled to the attacker's server and the malicious app does not need the Internet permission at all. The only limitation of this approach is that the maximum length of the query string is browser-specific. Different browsers have different limits for the length of their URLs. The discussion of Chromium[7] suggests that the length of character it can handle is 2Mb [5]. 2Mb is enough for smuggling about 6 full days' data if the malicious app generates and transmits one value (4 bytes per value) per second.

2.1 Challenges

We summarize major challenges to track a user with light sensor as follows.
1. Routes explosion. The area might be too big for the attacker to sample every distinct route in that area.

[7] Chromium is the open-source version of Chrome (the default browser of Android [1]).

2. Brightness level variations. The attacker might sample the routes, for example, with a different angle of phone placement or in a different weather or time of day, which has different environment brightness levels, than the victim.
3. Speed variations. The attacker might travel in a different speed from the user, for example, the speed differences between walking and bicycling.

We discuss the solutions to the first challenge in section 3.1 and section 3.3, the second challenge in section 3.3 and the third challenge in section 3.3.

3 Route Identification

The attack has two stages in total - off-line state and attack stage. The first stage includes off-line training and system building, the second stage is the attack.
Off-line Stage. This stage is shown in Fig. 1. During the off-line training step, the attacker collects the ambient light values of all the routes segments (see section 3.1) in an area using the attacker's own mobile phone. The attacker then trains the machine learning models and builds the tracking system on his server with the collected data. This step can be conducted without the victim's interaction and without data from the victim's mobile phone at all.
Attack Stage. This stage is shown in Fig. 2. The second stage is the attacking stage which needs the victim to have a malicious zero-permission app installed on his/her mobile phone (see section 2). The malicious app will send the victim mobile phone's ambient light sensor values (e.g., one value per second) via the Internet to a server under the attacker's control. The attacker can then use the values received from victim's mobile phone as the input to the system built in the first step to infer the route the victim travels and tracks the user.

3.1 Tracking User in an Area

When the user is on a long route with multiple junctions, there are two problems that we need to solve in order to successfully track the user. First, the total number of long routes would be large because of the combinations of multiple choices at each junction. Collecting data of all the long routes spanning multiple junctions will cost a lot of time in the off-line model training phase. Sometimes, it is even an impossible task for the attacker to collect data for every distinct long route if there are too many junctions (Challenge 1 in Section 2.1).

Second, some long routes might share the same route segments between junctions. For instance, a route starting from point $\boldsymbol{A}$ to point $\boldsymbol{C}$ and a route starting from point $\boldsymbol{A}$ to point $\boldsymbol{D}$ share the same route segment from point $\boldsymbol{A}$ to point $\boldsymbol{B}$ while they differ with route segment from point $\boldsymbol{B}$ to point $\boldsymbol{C}$ and route segment from point $\boldsymbol{B}$ to point $\boldsymbol{D}$. The shared segments between two long routes (e.g., segment $\boldsymbol{A}$ to $\boldsymbol{B}$) will lower the classification accuracy of two long routes (e.g., route $\boldsymbol{A}$ to $\boldsymbol{C}$ and route $\boldsymbol{A}$ to $\boldsymbol{D}$) due to the fact that the shared segments increases similarity of the two long routes and reduces their distinguishability.

In order to solve these two problems, instead of collecting the data from all the routes in an area to train the models to identify the routes, we sample

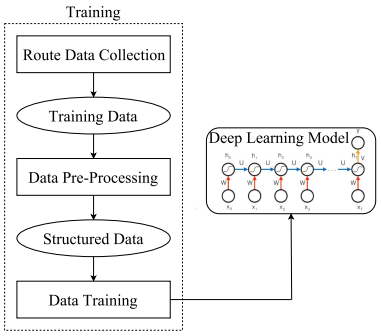


Fig. 1: Steps of Route Identification

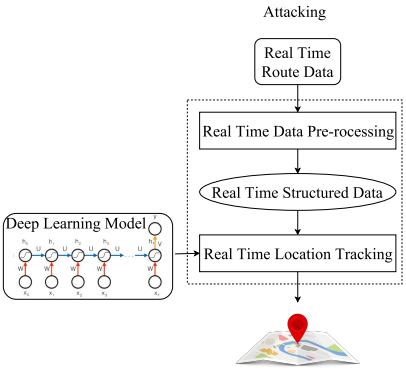


Fig. 2: Real-time Tracking System Overview

Fig. 3: An Example of Experiment Routes

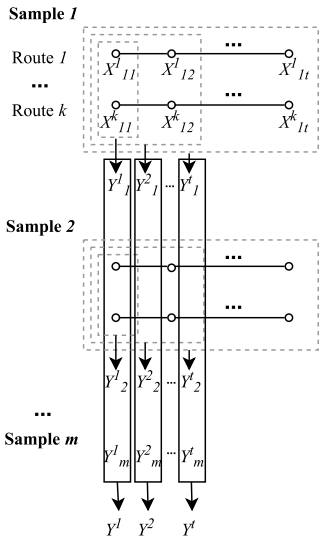


Fig. 4: Recomposition of Time Series

only the route segments separated by junctions. We collect the training datasets for each short route segment and train multiple models that identify different short route segments with the time series collected from them. Our system will automatically select the models that are used to identify the subsequent short route segments to fit the victim's data when the victim is at a junction. For more details, see 4.2.

In this way, we transfer the problem of identifying a long route into identifying several consecutive route segments separated by junctions. This significantly reduces the overheads in collecting the data in order to train the route identification models. Together with the approach we introduced in section 3.3, we solve the first challenge in section 2.1.

At the same time, this approach also solves the problem of low accuracies due to overlapping of route segments. Because the datasets for different long routes have overlapping route segments, while this approach has no overlapped route segments between each other.

Furthermore, it also greatly improves the scalability of our system. New route segments and new data can be easily added to our system by just training models for the new route segments without the needs to retrain all the models, because our approach tracks the victims on a long route by tracking the short route segments connected by junctions that the victim takes. Additionally, adding new routes does not decrease the accuracies of our system because the accuracy depends on individual route segments rather than the combined long route.

3.2 Data Collection for Route Identification

We select multiple route segments on campus to conduct our experiments. Fig. 3 shows one of the examples of the route segments from the same starting point to the same end point. For each route segment, we sample M times of the ambient light sensor values, which form the training datasets.

In order to capture all the possible brightness conditions where the victims might be travelling with, the M time samplings are carried out on different days with different weather and in different time of a day. The data is collected at a high data sampling rate of 100 times per second. Therefore, our datasets for the route segments cover the weather variations and different environment brightness due to different time of the day, which can reflect the real-world scenarios where the victim might be travelling in.

Suppose there are K route segments in our experiment areas, the final dataset we collect is $X = \{X^1, X^2, \ldots, X^K\}$ which contains K sub-datasets. The training datasets for route segment k can be denoted as $X^k = \{x_1^k, x_2^k, \ldots, x_m^k, \ldots, x_M^k\}$ $(1 \leq k \leq K, 1 \leq m \leq M)$, where $x_m^k = (x_{m1}^k, x_{m2}^k, \ldots, x_{mt}^k)$ is the time series with a length of t for the m-th sample of route segment k. Note that the length of each sample t is not a constant value, it might vary due to the route segment length and the speed of the traveler.

3.3 Data Preprocessing

The raw data we collected in the last step can not be applied in training the models directly. We need to perform the following four steps of preprocessing to the data in the datasets, namely normalisation, time series alignment, time series resampling, and speed resampling.

Normalization. The raw data is collected in different weather and times of the day. Therefore, the absolute values of light intensity might vary greatly. Moreover, even in the same weather, the range of the raw data we collect might be very wide, e.g., in the sun and in the shade. Normalizing the raw data removes the effects of weather and also speeds up the convergence of the training processes.

The normalization process is as follows. First, for each value in a time series (in a single sample), we subtract from it the mean value of all the values of that time series. Second, we divide the value derived in the first step by the standard deviation of all the values in that time series.

We highlight that the normalization to the datasets removes the absolute values of the light intensity and the numbers in the datasets after normalization

only reflect the light variation patterns along the routes which are not related to the absolute light intensity values. It thus removes the effects of phone placement angle, weather, time of day where the absolute values of the light intensity might be different. As long as it is the same route, the light variation patterns along that route will not change with different angles of phone placement, different weather and times of the day. Thus, the normalization to the datasets enables our models and systems to work on different angle of phone placement, different weather and time of the day, because they work on recognizing the light variation patterns along the routes rather than the absolute light values. The second challenge is resolved. Practicability and scalability of our approach are also greatly improved.

Time Series Alignment. When we are collecting data, we can not guarantee that every sample has exactly the same number of sampled values in a time series (for a single sample), i.e., same length in every time series. We can neither guarantee that the length of the time series collected from victim's mobile phone are exactly the same length as the ones in our training datasets. Therefore, we need to align the time series to make them the same length.

First, for the training datasets of the route segments, we find the longest time series in the training datasets and we record the length of this time series. For the rest of the time series in the training datasets, we repeat their last values until the lengths of them are the same as the length of the longest time series. After this step, all the time series of the routes in the training datasets are in the same length. Then, for the time series in testing datasets, if the time series are shorter than the length of the time series in its corresponding training datasets (same route), we again repeat the last value in that time series of the testing datasets until they have the same length as in their corresponding training datasets. If the time series in testing datasets are longer than the length of the times series in its corresponding training datasets, we do an average downsampling [26] to testing datasets making them same length as the corresponding training datasets.

After this step, the length of every time series in the training datasets and the testing datasets of a same route have the same length.

Time Series Resampling. We resample all the datasets at a sampling rate of one value per second. We take the median value of all the values within same second as the resampled ambient light sensor value for that second. Time series resampling benefits on our approach in the following three aspects. **First**, during the data collection, we have a high data sampling rate at 100 times per second. However, the values in one second might not change much and thus there are a lot of duplicated values in the collected datasets. The time series resampling help us speed up the training processes and it makes our model converge more quickly. **Second**, the cars or buses go by might introduce noises to the time series. Such noises sometimes might not last for the whole second. The resampling help us remove such noises in the data. **Third**, it enables the attacker to have more samples by resampling the sampled datasets. Thus, this can reduces the overhead of sampling the light sensors values of the route segments. It also improves the scalability of the attack and make it easier to launch this attack.

Speed Resampling. The training datasets collected by the attacker might have different speed from the victim's testing datasets, e.g., walking v.s. bicycling. In order to solve this challenge (Challenge 2 in Section 2.1), we perform resampling to the training datasets to get different datasets representing different speeds. We use the speed in the original training datasets as our baseline speed, then we perform resampling to the training datasets to get the datasets for different speeds. In our models, we resample the datasets into two new datasets. Therefore, we have in total 3 different datasets for training different models. We let the speed of our original datasets to be v. The speeds of the three different datasets are $0.5v$, v and $2v$ datasets.

For the $0.5v$ datasets, we repeat the value in one second in the original datasets into two seconds (the original datasets have been data preprocessed, which has one value per second). For example, one of the time series of the original datasets is $X = \{x_1, x_2, \ldots, x_t\}$ where t is the length of the time series of t seconds. Then, after speed resampling into $0.5v$, the time series becomes $X' = \{x_1, x_1, x_2, x_2, \ldots, x_t, x_t\}$. In this way, the length of the time series is two times of the original one but the route length is the same. Thus, the speed is reduced by half to be $0.5v$. Similarly, to get the datasets of $2v$, we get one value every two seconds from the original datasets. We resample the original time series $X = \{x_1, x_2, x_3, x_4 \ldots, x_{t-1}, x_t\}$ into $X' = \{x_2, x_4, \ldots, x_t\}$. The length of the time series is half of the original time series but the route length is still the same. Thus, the speed becomes two times of the original speed ($2v$). We highlight that we use a high sampling rate for the data collection (100 times per second). Therefore, the resampling does not lose much "information" of routes.

Therefore, after this step, there are different datasets for a route segment corresponding to different speeds. The datasets for the same speed of different route segments starting from the same junctions have the same length. Therefore, we have solved the third challenge in section 2.1. It also enables our models to track victims on different travelling patterns, e.g., walking, running, biking etc.

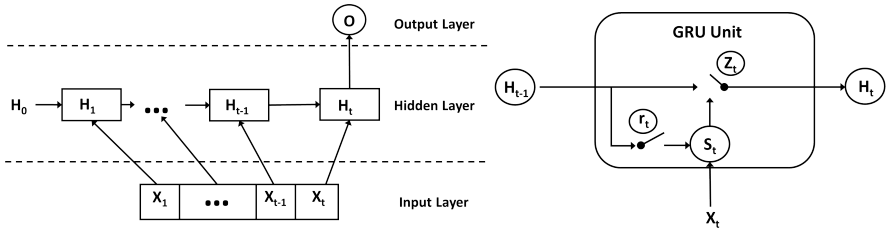
3.4 RNN Model with a Gated Recurrent Unit

We employ Recurrent Neural Network (RNN) [11,28] with a Gated Recurrent Unit (GRU) [30,32] as the engine of our route identification models and real-time location tracking system. RNN is one of the most popular deep learning methods for analyzing sequence information, especially time series data. An RNN module consists of an input layer, one or multiple hidden layers and one output layer [13]. The current output of the RNN module not only depends on the current input, but also depends on the previous computations from previous inputs. It performs the same calculation for every element of a sequence (time series in our case). The structure of a one-hidden-layered RNN is shown in Fig. 5a. The calculations of the hidden units and the output units iterate over all elements in a time series with the following equations:

$$h_t = tanh(Ux_t + Wh_{t-1} + b) \tag{1}$$

$$o_t = tanh(Vh_t + c) \tag{2}$$

where *tanh* is an activation function [18,16], U, W and V are corresponding parameters that need to be trained.



(a) one-hidden-layer RNN module (b) Structure of a GRU

In practice the output of a standard RNN can only make use of the information in a few steps back due to the vanishing gradient problem [8,27]. In order to learn longer-term dependencies, we change the way in calculating the hidden units by using Gated Recurrent Units (GRUs). The calculation structure of a hidden unit h_t of GRU is shown in Fig. 5b, where h_t is computed based on x_t and h_{t-1} with a gating mechanism. The reset gate r_t in GRU determines how to combine the new input x_t with the previous memory h_{t-1} in computing s_t, where s_t is a "candidate" hidden state. The calculation of h_t uses the update gate z_t to define how much of the previous memories to keep around.

Detailed equations for calculating a GRU hidden unit are shown as follows:

$$z_t = \sigma(x_t U^z + h_{t-1} W^z + b^z) \tag{3}$$
$$r_t = \sigma(x_t U^r + h_{t-1} W^r + b^r) \tag{4}$$
$$s_t = tanh(x_t U^s + (h_{t-1} \odot r_t) W^s + b^s) \tag{5}$$
$$h_t = (1 - z_t) \odot s_t + z_t \odot h_{t-1} \tag{6}$$

where the function $\sigma(x) = max(0, min(1, x * 0.2 + 0.5))$ is the hard sigmoid and $\odot$ denotes component-wise multiplication.

Overall, our route identification algorithm first performs data preprocessing on the raw time series data and then the preprocessed data is fed into the deep learning module to train proper models. Given a testing route sample under classification, the outputs of the RNN module provide K probabilities with each showing the probability of the current route sample being route i (i from 1 to K is a route whose data has been collected as training data). We then decide a most likely route the testing sample to be by the one with highest probability.

3.5 Models Training

We use the route segments between point ***A*** and point ***B*** in Fig. 6a as an example to demonstrate the model training process. There are three route segments - route segment 4, route segment 5 and route segment 6, respectively. For the

datasets of the route segments starting from same junction ($\boldsymbol{A}$) in the same speed, let us say their lengths of each time series after data preprocessing are t. That means there are t seconds with t values in every time series of every route segment starting from point $\boldsymbol{A}$ to point $\boldsymbol{B}$ after data preprocessing. We thus train t models, each of which identifies 3 classes (one class per route segment). The first model accepts time series of length 1, the second model accepts time series of length 2, and so on until the t-th model accepts time series of length t. The overall process of this example is shown in Fig. 4

Suppose that there are M samples for each route segment starting from point $\boldsymbol{A}$ to point $\boldsymbol{B}$. The time series of the first sample of route segment 4 is $X_1^4 = \{X_{11}^4, X_{12}^4, \ldots, X_{1t}^4\}$; the time series of the first sample of route segment 5 is $X_1^5 = \{X_{11}^5, X_{12}^5, \ldots, X_{1t}^5\}$ and the time series of the first sample of route segment 6 is $X_1^6 = \{X_{11}^6, X_{12}^6, \ldots, X_{1t}^6\}$. We follow the same notation conventions as in Section 3.2 where the k of X_m^k denotes the route segment ID, m denotes the m-th sampling to the route segment. Similarly to notation in Section 3.2, t in X_{mt}^k is the length of the longest time series after data preprocessing. Then, we let $Y_1^1 = \{X_{11}^4, X_{11}^5, X_{11}^6\}$, $Y_1^2 = \{\{X_{11}^4, X_{12}^4\}, \{X_{11}^5, X_{12}^5\}, \{X_{11}^6, X_{12}^6\}\}, \ldots$, $Y_1^i = \{\{X_{11}^4, X_{12}^4, \ldots, X_{1i}^4\}, \{X_{11}^5, X_{12}^6, \ldots, X_{1i}^6\}, \{X_{11}^7, X_{12}^7, X_{1i}^7\}\}$, $\ldots$, $Y_1^t = \{X_1^4, X_1^5, X_1^6\}$ $(1 \leq i \leq t)$. The set Y_1^i $(1 \leq i \leq t)$ represents the time series of the first sample of length i for all the route segments starting from point $\boldsymbol{A}$ to point $\boldsymbol{B}$ in the same speed. Similarly, we can derive set $Y_m^i = \{X_m^4, X_m^5, X_m^6\}$, $(1 \leq m \leq M, 1 \leq i \leq t)$ which represents the time series of the m-th sample of length i for all the route segments starting from point $\boldsymbol{A}$ to point $\boldsymbol{B}$. Finally, we have $Y^1 = \{Y_1^1, Y_2^1, \ldots, Y_M^1\}$, $Y^2 = \{Y_1^2, Y_2^2, \ldots, Y_M^2\}$, $\ldots$, $Y^i = \{Y_1^i, Y_2^i, \ldots, Y_M^i\}$ and so on until $Y^t = \{Y_1^t, Y_2^t, \ldots, Y_M^t\}$ $(1 \leq i \leq t)$. Each of such set Y^i is of length M (the total number of sampling for each route segment). Finally, for each dataset of Y^i where i ranges from 1 to t, we train a model that accepts length i time series (where i ranges from 1 to t) and identifies (or classifies) route segments 4, 5 and 6 starting from point $\boldsymbol{A}$ at the same speed. We repeat the above process for every junction in both directions.

In summary, after this training step, we have trained t models for every speed at every junctions in both directions, where t is the length of the longest time series among all the samples for the route segments sharing the same starting junction after data preprocessing.

4 Applying Light Sensor in Real-time Tracking

The fact that light sensor is giving out user's location information can even be applied to track the user in real-time if user's starting location is known. It is a matter of time that the attacker can eventually build a system using the light sensor values to track the users in real-time whose accuracy is comparable to use GPS. In the following, we take the first step towards this goal trying to apply light sensor values in real-time tracking the users. Our system shows that this goal is practical and feasible.

The overall flow of our system is shown in Fig. 2. The left-hand side are the route identification models trained in Section 3. The right-hand side is the actual attacking. With the real-time values from the ambient light sensor of the victim's mobile phone, our system first preprocesses the data. Then the preprocessed structured data is fed into the off-line trained models in attacker's server. Finally, based on the trained models, our system outputs the route segments that the victim is on and infer the victim's location in real-time.

4.1 Real-time Location Estimation

At the attacking steps (shown in Section 2), the data from the victim's mobile ambient light sensor is transmitted to attacker's server in real-time. The attacker performs normalization and time series resampling in the same way as in Section 3.3. Then, the preprocessed datasets are sent to our models for identifying.

In our training datasets, the GPS locations are also recorded. Note that after the data preprocessing steps in the model training stage, all the time series for route segments starting from the same junctions in the same direction with the same speed have the same length. The average GPS location for each second is calculated over all the GPS locations at that same second in training datasets. For example, let $G_m^k = \{g_{m1}^k, g_{m2}^k, \ldots, g_{mt}^k\}$ represent the GPS locations of route segment k from the first second to the t-th second at the m-th sample in the training datasets, where $g_{mt}^k = \{g_{m(t-1)r+1}^k, g_{m(t-1)r+2}^k, \ldots, g_{mtr}^k\}$ and r is the sampling rate meaning that there are r samples in a second. Recall that the sampling rate of our datasets is 100 times per second. Thus in our case, $r = 100$.

We let $\overline{g_{mt}^k}$ be the median value of all the values in the set g_{mt}^k, and we use it as the average GPS location for the value of the t-th second for route segment k of the m-th sample. Therefore, the set of average locations at the m-th sample of route segment k can be represented as $\overline{G_m^k} = \{\overline{g_{m1}^k}, \overline{g_{m2}^k}, \ldots, \overline{g_{mt}^k}\}$. Remark that after the data preprocessing, all the samples have the same length t (the length of the longest time series) and it is the same for every sample from 1 to M. After that, we let $F_t^k = \{\overline{g_{1t}^k}, \overline{g_{2t}^k}, \ldots, \overline{g_{Mt}^k}\}$ be the set of locations at the t-th second of all M samples for route segment k. The average location at the t-th second of all M samples for route segment k is thus $\overline{F_t^k}$ which is the average value of all the elements in the set F_t^k. Finally, we can have the set of average locations for a route segment k across all M samples for every second which is $\overline{G^k} = \{\overline{F_1^k}, \overline{F_2^k}, \ldots, \overline{F_i^k}, \ldots, \overline{F_t^k}\}$, $(1 \leq i \leq t)$.

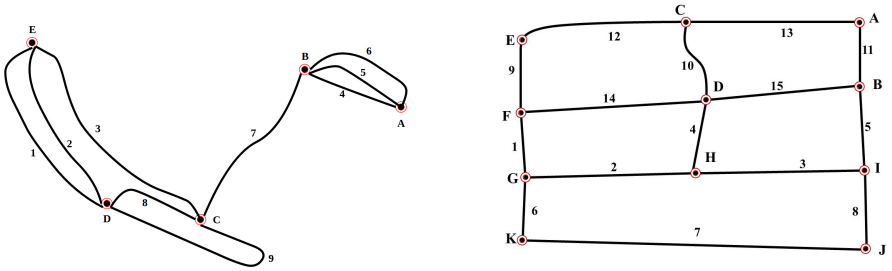
If there are t values after real-time data preprocessing at the attacking stage, the t values are feed into the model that accepts length t time series. The model then outputs the identification result indicating the victim is currently on which route segment. The result means that the victim is on that identified route segment and she has traveled for t seconds. Thus, our real-time tracking system outputs the average GPS location at t-th second of the identified route segment.

4.2 Handover at Junction

When the victim is tracked until s/he reaches the junction, based on the victim's history (previous traveled route segments and directions), the system resets the time series to length zero and automatically switches to the models of the route segments starting from that junction to identify the new route segment using the new preprocessed time series from the victim's mobile phone. For example, if the victim is travelling on route segment 3 from junction $\boldsymbol{E}$ to junction $\boldsymbol{C}$ as shown in Fig. 6a. When the victim has come to the end of route segment 3, our system automatically uses the models that identifies (classifies) route segments 7, 8 and 9 to fit the new values coming from victim's light sensor. Then based on the outputted results of the models, the system selects the new route segment which best fits the new light sensor values and outputs the estimated real-time locations of the victim. In this way, the long route the victim travels which spanning over multiple junctions can be identified accurately and efficiently by identifying multiple connected route segments.

5 Evaluation

We have collected two sets of data in two areas in two different cities at different latitudes. One area is near the Equator and the other is in the Northern Hemisphere at temperate zone. We refer to the first and second datasets as dataset $\boldsymbol{Q}$ ($\boldsymbol{Q}$ area) and dataset $\boldsymbol{T}$ ($\boldsymbol{T}$ area) respectively in the rest of this section. The map of these two areas are shown in Fig. 6a and Fig. 6b.



(a) Area Near Equator ($\boldsymbol{Q}$) (b) Area in Temperate Zone ($\boldsymbol{T}$)

In Fig. 6a, the joining point of lines represents a junction. There are 5 junctions from $\boldsymbol{A}$ to $\boldsymbol{E}$. Between two adjacent junctions, there are multiple route segments represented by solid lines. There are in total 9 route segments in the area separated by the 5 junctions. Each route segment is labelled with a route segment ID which ranges from 1 to 9. The total combinations of long route can be up to 15 without circle from point $\boldsymbol{A}$ to point $\boldsymbol{E}$. The datasets for this area are collected by two persons. One of them samples the routes segments from 1 to 9, and the other samples the data for the long routes in a different speed from

the first person for testing our real-time location tracking system. The datasets for the route segments are collected using the approach we introduced in Section 3.2. Every route segment has been sampled at least for 30 times for each direction and the long routes are sampled once. The datasets are collected using a Nexus 5 mobile phone with a bicycle.

For the area in Fig. 6b, there are totally 11 junctions with 15 route segments. Each of the route segment is assigned a route segment ID ranging from 1 to 15. There are already 18 different long routes for a user to travel from junction $\boldsymbol{A}$ to junction $\boldsymbol{K}$ without going backwards. The total combinations of the long routes that a user can take are too many for the attacker if he wants to sample every long route to train the models. Similar to the datasets in the area of Fig. 6a, there are two persons collecting data for the area in Fig. 6b. One collects all the data for route segments from 1 to segments 15, and the other collects datasets once for randomly selected long routes used for testing our real-time tracking system. Every route segment is sampled at least 30 times for each of the direction. For the sake of diversity, we use another Nexus 5 mobile phone, which is different from the one that is used to collect data for $\boldsymbol{Q}$ area, to sample the data in area $\boldsymbol{T}$ with bicycles. We summarize the statistics of our datasets in Table 1.

5.1 Route Segments Identification Evaluation

The datasets are first preprocessed and then be fed into the machine learning models for training and testing. We set the number of hidden unit of our RNN models as 10 in our experiments and the number of input unit as 1 because we only fit the model with the values from light sensor (one feature used). We apply a 2-hidden-layer GRU module to capture higher-level feature interactions between different time steps.

We further apply 5-fold cross validation in estimating how accurately the machine learning models we trained perform in practice. The datasets are randomly partitioned into 5 similar sized sub-samples. This process is repeated 5 times, with each of the 5 sub-samples used once in each validation as the validation data and we use the remaining 4 sub-samples as the training data. The accuracies of all the testing results from the validations are shown in Table 2.

The Start and End for accuracies of $\boldsymbol{Q}$ area (Table 2a) specify the travelling directions of the victim. For example, the accuracy in Table 2a in the first row with starting junction to be $\boldsymbol{A}$ and ending junction to be $\boldsymbol{B}$ means that the victim starts travelling at junction $\boldsymbol{A}$ and travels towards $\boldsymbol{B}$ and the victim has 3 route segments to choose to travel on. The accuracy for identifying these three route segments is 99.2%. In Table 2b, the start column specifies the starting junction of the victim and the number of routes specifies how many route segments that he/she can travel on. The overall result in Table 2 shows that the route segment identification is very accurate. The accuracy shows our approach is effective. The results also prove that the light sensor values can be used to differentiate and identify route segments. As the route identification evaluation shows, we have achieved the first goal proposed in Section 1.

Table 1: The Datasets in Two Areas

(a) Summaries for $\boldsymbol{Q}$ Area and $\boldsymbol{T}$ Area

Area	$\boldsymbol{Q}$	$\boldsymbol{T}$
# Segments	9	15
# Junctions	5	11
# Volunteers	2	2
# Sampled Times	60	60

(b) Route Segments Length for $\boldsymbol{Q}$ Area and $\boldsymbol{T}$ Area

Area / **ID**	Length in m	
	$\boldsymbol{Q}$	$\boldsymbol{T}$
1	248	71
2	230	90
3	292	50
4	131	70
5	132	70
6	177	76
7	155	140
8	123	77
9	285	92
10	N.A.	96
11	N.A.	90
12	N.A.	83
13	N.A.	63
14	N.A.	90
15	N.A.	57

Table 2: Route Segments Identification

(a) Accuracies for Area $\boldsymbol{Q}$

Start	**End**	**#Routes**	**Accuracy%**
A	B	3	99.2
B	A	3	96
C	D	2	98.8
D	C	2	98.8
D	E	2	100
E	D	2	96.3

(b) Accuracies for Area $\boldsymbol{T}$

Start	**#Routes**	**Accuracy%**
A	2	91.7
B	3	83.3
C	3	94.4
D	4	79.2
E	2	91.7
F	3	88.9
G	3	72.2
H	3	83.3
I	3	72.2
J	2	100.0
K	2	66.7

5.2 Real-time Location Tracking Evaluation

Fig. 7 shows four results of the estimation errors of the long route experiments. Fig. 7a to Fig. 7d show the system predictions and the estimation errors in meters of the long route experiments. The vertical red solid lines in Fig. 7 show us the time when the user actually reaches the end of a route segment and a junction, while the vertical green dashed lines show us the system's prediction of the time that the victim reaches the junction. The real-time location estimation errors, i.e., the distance between the user's real location and the predicted location are also shown in Fig. 7 with blue solid line. For example, in Fig. 7a, our system predicts that the user has arrived at the first junction at about 62nd second while the user actually reaches the first junction at about the 79th second. Our system prediction result leads the actual result roughly by 17 seconds. The largest estimation error shown in Fig. 7a is about 140 meters while the average estimation errors is much lower at about 40 meters to 50 meters.

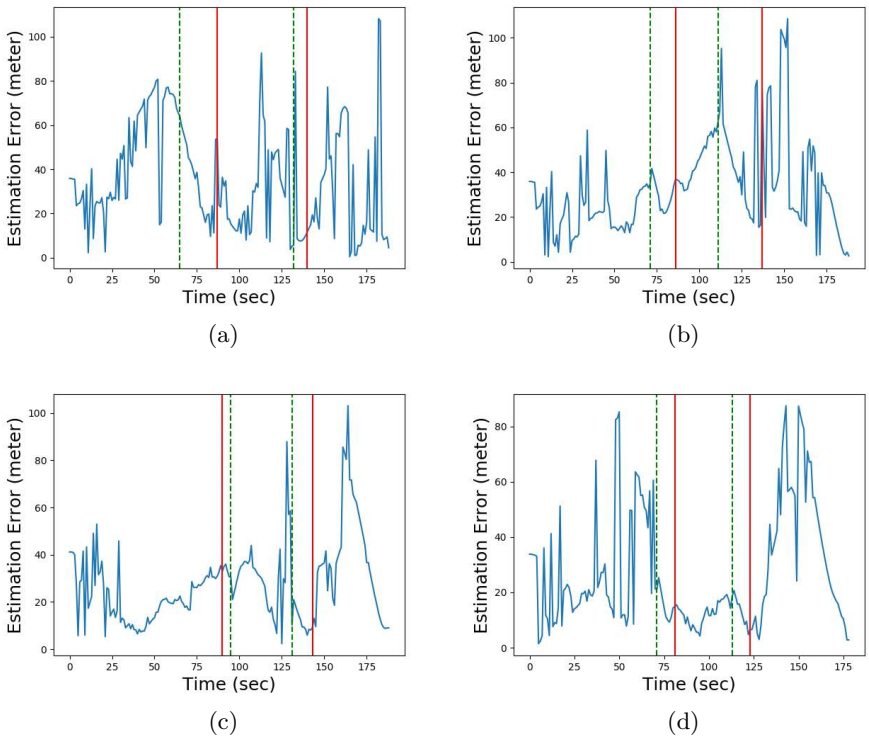
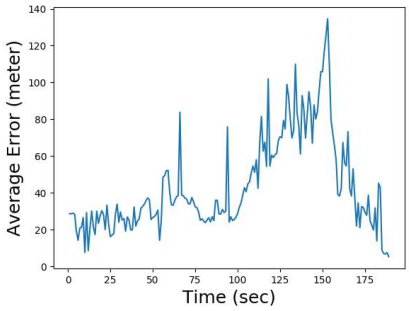


Fig. 7: Estimation Errors for the Long Routes

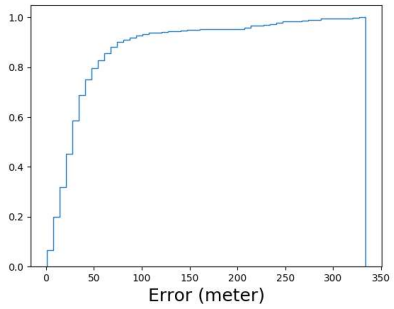
The overall accuracies in Fig. 7a, Fig. 7b, Fig. 7c and Fig. 7d are 75.66%, 66.67%, 58.73% and 53.16% respectively. The average accuracy for the long route experiments in Fig. 7 is 63.56%. This means that about 63.56% of the time (e.g. 63 seconds out of 100 seconds), the system can locate the user on the correct route. Fig. 8 shows the average and distribution of errors for Fig. 7. Fig. 8a shows the average estimation error of the four estimation errors from Fig. 7a to Fig. 7d. On the whole, the trend of the average error is quite stable. The overall average error is around 70 meters. From Fig. 8b and Fig. 8c, we can see that 80% of the errors is less than 50 meters. From Fig. 7a to Fig. 7d we know that the times of user that reaches the end of the first route segment is around 70th second to 80th second and the time that the user reaches the second route segment is around 130th second to 140th second.

6 Related Work

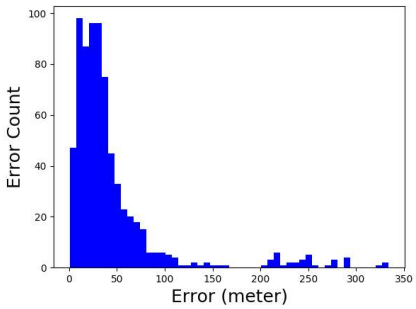
Most of the previous discovered side channels on mobile platform are motion related sensors (i.e., gyroscope, accelerometer). Our work is most related to



(a) Average Error



(b) Error Cumulative Distribution



(c) Error Histogram

Fig. 8: Average and Distribution Errors for the Long Routes

PowerSpy [21]. In PowerSpy, the authors discover that the power consumption speed can be used to infer user's location. Similarly, we both identify a side channel that leaks user's location information. Unlike [21], this paper uses a different side channel and algorithm to tell the location in a more fine-grained scale (e.g., within 70m). Moreover, we have enhanced our approach to perform a random route real-time location tracking while in [21], it only reports real-time location tracking on one specified known route with only estimation error. Our evaluation, which is similar to the one in [21] shows that our approach achieves a higher accuracy in both route identification (91% v.s. 85%) and real-time user tracking (64%), and lower estimation error (∼70m v.s. ∼1km). The work in Elastic Pathing [9] also presents a side channel for mobile users. They propose to use merely speed information to track the route of the mobile users. However, different from this work, it is based on a different attack model and algorithm. In addition, our system achieves much higher accuracies (91% v.s. 26%) while much lower estimation errors (∼70m v.s. ∼500m).

The following works [24,22,12] use motion sensors such as gyroscope, accelerometer, to infer the route of a user. They also have similar techniques. First, by using the motion sensors to infer the trajectory or shape of the route

the user travels. Then, assuming the city of the user is known, they try to find the same shape of route from the maps and infer the route from the matches. These works differ in the techniques to matches the shape of route inferred from the motion sensors to the shape of route on maps. While our approach is significant different from them in that we only rely on the ambient light sensor which itself is totally not designed for helping navigation or tracking after user. More importantly, light sensor itself does not reveal any information on user's motion while accelerometer or gyroscope in some sense reveals user's motion.

In this paper [33], the authors discover a side channel using mobile phone's microphone. When user is using voice navigation, the length of the speech can be used to infer the navigation information and thus infer the route the user travels. Different from our work, this work focus on inferring the secrets from the public resources of Android while our work mainly presents and proves that light sensor is indeed a novel side channel that is leaking mobile user's location information. In the paper [29], the authors using innocuous permissions to steal sensitive information from audio sensor. This paper [7], the authors using mobile phone's camera and microphone to fingerprinting the locations. Unlike the above two paper, we do not rely on sensor permissions and we can identify not only the routes the user travels but also track the user location on that route.

7 Conclusion

This work presents a novel side channel of ambient light that can be used to track the mobile users. We developed a location tracking system that can identify the routes the user travels and tracks him/her in real-time with the ambient light sensor. The experiments show that our tracking system works at high accuracy and outperform other similar works.

Acknowledgement

This paper is supported by the National Research Foundation, Prime Minister's Office, Singapore under its National cybersecurity R&D Program (TSUNAMi project, Award No. NRF2014NCR-NCR001-21) and administered by the National Cybersecurity R&D Directorate, and the National Natural Science Foundation of China (No.61702045).

References

1. The chromium projects, https://www.chromium.org/
2. Normal permissions, https://developer.android.com/guide/topics/permissions/normal-permissions.html
3. Requesting permissions at run time, https://developer.android.com/training/permissions/requesting.html
4. Stroage options, https://developer.android.com/guide/topics/data/data-storage.html#AccessingExtFiles

5. Loading large urls kills the renderer (Jan 2011), https://bugs.chromium.org/p/chromium/issues/detail?id=69227
6. Why location privacy matters (Feb 2017), https://thetinhat.com/blog/thoughts/location-privacy.html
7. Azizyan, M., Constandache, I., Roy Choudhury, R.: Surroundsense: mobile phone localization via ambience fingerprinting. In: Proceedings of the 15th annual international conference on Mobile computing and networking. pp. 261–272. ACM (2009)
8. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5**(2), 157–166 (1994)
9. Gao, X., Firner, B., Sugrim, S., Kaiser-Pendergrast, V., Yang, Y., Lindqvist, J.: Elastic pathing: Your speed is enough to track you. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 975–986. ACM (2014)
10. Goodin, D.: Yahoo says half a billion accounts breached by nation-sponsored hackers (Sep 2016), https://arstechnica.com/information-technology/2016/09/yahoo-says-half-a-billion-accounts-breached-by-nation-sponsored-hackers/
11. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: Draw: A recurrent neural network for image generation. arXiv preprint arXiv:1502.04623 (2015)
12. Han, J., Owusu, E., Nguyen, L.T., Perrig, A., Zhang, J.: Accomplice: Location inference using accelerometers on smartphones. In: Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on. pp. 1–9. IEEE (2012)
13. Haykin, S.: Neural networks: a comprehensive foundation. Prentice Hall PTR (1994)
14. Help, G.A.: About targeting geographic locations, https://support.google.com/google-ads/answer/2453995?hl=en
15. Iqbal, M.U., Lim, S.: Privacy implications of automated gps tracking and profiling. IEEE Technology and Society Magazine **29**(2), 39–46 (2010)
16. Karlik, B., Olgac, A.V.: Performance analysis of various activation functions in generalized mlp architectures of neural networks. International Journal of Artificial Intelligence and Expert Systems **1**(4), 111–122 (2011)
17. Lawler, R.: Equifax security breach leaks personal info of 143 million us consumers, https://www.engadget.com/2017/09/07/equifax-hack-143-million/
18. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural networks **6**(6), 861–867 (1993)
19. Levy, K.: Here's what happens when cellphone users get their own lane on the sidewalk (July 2014), https://www.businessinsider.com/cellphone-walking-lane-2014-7/?IR=T
20. Masinter, L., Berners-Lee, T., Fielding, R.T.: Uniform resource identifier (uri): Generic syntax. RFC 3986 (2005), https://tools.ietf.org/html/rfc3986
21. Michalevsky, Y., Schulman, A., Veerapandian, G.A., Boneh, D., Nakibly, G.: Powerspy: Location tracking using mobile device power analysis. In: USENIX Security Symposium. pp. 785–800 (2015)
22. Narain, S., Vo-Huu, T.D., Block, K., Noubir, G.: Inferring user routes and locations using zero-permission mobile sensors. In: Security and Privacy (SP), 2016 IEEE Symposium on. pp. 397–413. IEEE (2016)
23. Nasar, J.L., Troyer, D.: Pedestrian injuries due to mobile phone use in public places. Accident Analysis & Prevention **57**, 91–95 (2013)

24. Nawaz, S., Mascolo, C.: Mining users' significant driving routes with low-power sensors. In: Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems. pp. 236–250. ACM (2014)
25. Olmstead, K., Atkinson, M.: Apps permissions in the google play store (2015), http://www.pewinternet.org/2015/11/10/an-analysis-of-android-app-permissions/
26. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. Computer Vision–ECCV 2006 pp. 568–580 (2006)
27. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning. pp. 1310–1318 (2013)
28. Sak, H., Senior, A., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
29. Schlegel, R., Zhang, K., Zhou, X.y., Intwala, M., Kapadia, A., Wang, X.: Soundcomber: A stealthy and context-aware sound trojan for smartphones. In: NDSS. vol. 11, pp. 17–33 (2011)
30. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: EMNLP. pp. 1422–1432 (2015)
31. Whittaker, Z.: 198 million americans hit by 'largest ever' voter records leak. http://www.zdnet.com/article/security-lapse-exposes-198-million-united-states-voter-records/ (Jun 2017)
32. Wu, Z., King, S.: Investigating gated recurrent networks for speech synthesis. In: Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. pp. 5140–5144. IEEE (2016)
33. Zhou, X., Demetriou, S., He, D., Naveed, M., Pan, X., Wang, X., Gunter, C.A., Nahrstedt, K.: Identity, location, disease and more: Inferring your secrets from android public resources. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 1017–1028. ACM (2013)