Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 5153265, 15 pages
https://doi.org/10.1155/2018/5153265

Research Article

SSID Oracle Attack on Undisclosed Wi-Fi Preferred Network Lists

Ante Dagelić,[1] **Toni Perković**,[2] **Bojan Vujatović**,[1] **and Mario Čagalj**[1]

[1]*FESB, University of Split, Split, Croatia*
[2]*University Department of Forensic Sciences, University of Split, Split, Croatia*

Correspondence should be addressed to Toni Perković; toperkov@unist.hr

Received 15 March 2018; Revised 7 June 2018; Accepted 8 July 2018; Published 22 July 2018

Academic Editor: Mauro Femminella

User's location privacy concerns have been further raised by today's Wi-Fi technology omnipresence. Preferred Network Lists (PNLs) are a particularly interesting source of private location information, as devices are storing a list of previously used hotspots. Privacy implications of a disclosed PNL have been covered by numerous papers, mostly focusing on passive monitoring attacks. Nowadays, however, more and more devices no longer transmit their PNL in clear, thus mitigating passive attacks. Hidden PNLs are still vulnerable against active attacks whereby an attacker mounts a fake SSID hotspot set to one likely contained within targeted PNL. If the targeted device has this SSID in the corresponding PNL, it will automatically initiate a connection with the fake hotspot thus disclosing this information to the attacker. By iterating through different SSIDs (from a predefined dictionary) the attacker can eventually reveal a big part of the hidden PNL. Considering user mobility, executing active attacks usually has to be done within a short opportunity window, while targeting nontrivial SSIDs from user's PNL. The existing work on active attacks against hidden PNLs often neglects both of these challenges. In this paper we propose a simple mathematical model for analyzing active SSID dictionary attacks, allowing us to optimize the effectiveness of the attack under the above constraints (limited window of opportunity and targeting nontrivial SSIDs). Additionally, we showcase an example method for building an effective SSID dictionary using top-N recommender algorithm and validate our model through simulations and extensive real-life tests.

1. Introduction

Location privacy presents one of the most challenging problems in today's mobile era. Modern mobile devices such as smartphones, tablets, or smart watches collect information from surrounding devices that can be used for Wi-Fi-based positioning systems [1]. In a similar fashion, these devices emit radio signals that can be used for localization and tracking by using, for example, cell tower trilateration [2]. Furthermore, Wi-Fi information transmitted from mobile devices can be used for indoor tracking and targeted services [3–6]. Many of these solutions base their services upon collecting a number of management frames (e.g., beacon, probe request, and probe response frames,) that are transmitted by devices, whose main purpose is to establish a fast and efficient connection during the authentication process.

A number of papers discuss various threats to user's privacy based upon collected probe requests containing an Access Point's (AP) Service Set Identifier-SSID [7–13]. Wi-Fi-enabled devices using Active Service Discovery broadcast these probe request frames aimed at the set of preferred APs to increase the connection speed. Such probe request packet contains an SSID of a previously associated AP stored in the device's preferred network list-PNL (a complete list of previously associated APs), the MAC address of the mobile device, and some other information. Simply by observing these probe request packets, a potential attacker can learn a complete PNL, which raises serious privacy threats. As an example, we highlight `SSID=Shelbourne Medical Clinic` from a large amount of Wi-Fi traffic that we passively collected during a festival that attracts 50 000 people. This problem is highlighted by the fact that SSIDs can also be geolocated (e.g., by using a WIGLE service [14]) thus violating user's location privacy.

To mitigate these privacy breaches made possible by active scanning (AS), the devices could use passive scanning

(PS) technique whereby devices transmit no probe request packets but instead listen to beacon packets emitted from the APs. Unfortunately, compared to active scanning, this technique accomplishes slower connection times [15]. Besides direct probe mode, active scan also supports a broadcast probe mode where probe request packets hold an empty SSID field. All neighboring APs (including the ones in the device's PNL) will answer the broadcast probe request. This mode is widely used in most of today's Android-based mobile devices (Although a recent discovery showed that some Android devices transmit their PNL while being in low-power state with their screen turned off, it seems that Google was unaware of such vulnerability [16, 17]). Using this mode of operation, the attacker cannot learn more information about the user's location history by simply observing broadcast probe request packets.

However, the mentioned broadcast probe mode does not protect the user from an active attack. One of the simplest methods to verify whether a user was connected to an AP in a specific location includes the creation of a rogue AP with that specific name (SSID) and identifier, i.e., by mounting a so-called Karma attack [18], verifying if the victim initiates a connection to it. In the recent work where Karma attack was performed, only a fraction of different SSIDs were considered, by using a dictionary containing a small number of the most popular APs that are found in the surrounding area (a popular café bar or a restaurant, etc.) [19]. Similarly, faking an AP has also been used in [20] by performing a simple dictionary attack on hidden SSID networks, using a Phyton library Scapy. However, as opposed to static environments in which users are at their homes and offices with laptops and desktop PCs, users with smartphones are usually mobile with their devices; hence the opportunity window for the attack can be very time limited. In contrast to existing work, in this paper we focus on greatly improving the attack's performances. We want to find methods and techniques that will allow the attacker to query every SSID from as large as possible dictionary within the given opportunity window against a victim's device. Moreover, we are interested in discovering *nontrivial* SSIDs, thus increasing the importance of disclosed SSIDs from user's PNL.

Such form of a dictionary attack is universal to all devices that implement IEEE 802.11 standard, regardless of the active or passive scanning technique. The following contributions have been made:

(i) We developed an analytical model for the SSID dictionary attack;

(ii) We created a probing strategy that minimizes probing time of the complete SSID dictionary;

(iii) We carried out extensive real-life tests and simulations to validate our model.

Additionally, an example SSID dictionary building algorithm is provided using a modified top-N recommendation algorithm [21] to reflect victim's PNL.

The rest of the paper is organized as follows: in Section 2 we are defining the general attack setup and describing device behavior. Section 3 provides a comprehensive mathematical model for the attack strategy followed by Section 4, which provides an overview of the practical tests carried out on multiple devices alongside performed simulation and an example algorithm for building the dictionary. Finally, we cover related work and conclude in Sections 5 and 6.

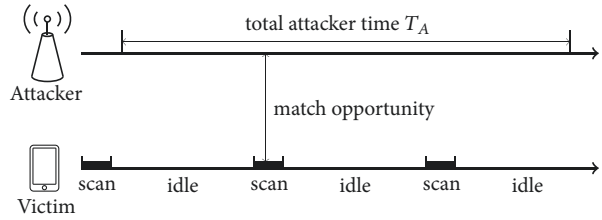

Figure 1: SSID Oracle attack basic setup.

2. System and Attacker Model

Regardless of the type of scanning technique (Active Service Discovery with or without an empty probe or Passive Service Discovery), in our model we assume that devices are periodically scanning for available APs as shown in Figure 1, whereby the scanning period is significantly shorter than the idle period when the device is not scanning for the available networks. The scan period depends on the device model, OS, settings controlled by the user, such as keeping Wi-Fi on/off during sleep mode [22], location services, and also general sensors' activity (accelerometer, gyroscope), which might change the state of the device. Other features that can affect the periodic scanning behavior of Wi-Fi devices are determined by the activity of various services, such as if cellular data is turned on, or if the smartphone is in maintenance mode, or if the user is extensively using Internet. A comprehensive overview of measured scanning periods depending on the mentioned settings and scenarios can be found in [23].

2.1. Periodic Wi-Fi Scan Intervals. Scanning periods in practice do change in length, either in fixed increases or exponentially. However, the scanning period tends towards repetition in the same length intervals, as can be seen from [23, 24]. The authors have tested various Apple and Android devices and the results have shown that all the devices end up scanning for Wi-Fi networks in fixed intervals. Periodic behavior was observed in various use cases for the victim (Wi-Fi settings screen, other screens, and display off). The authors in [23] have also concluded that these are fixed times depending on the victim's device and operating system, which has also been observed in our tests. Furthermore, the periodical behavior can be actively provoked, and the required scanning/idle time durations can be measured by the attacker, as it will be described in the following subsection. Increasing the scanning frequency also increases the success rate of various Wi-Fi attacks/data analysis and has been noted by others before [25].

2.2. Increasing the Scanning Frequency. Although the model presented in this paper optimizes the performances for any given periodic scanning frequency, there are ways for the

attacker to actively control the periodic behavior of Wi-Fi scanning intervals. For example, in our tests with smartphone devices, we showed that if the device is in sleep mode having cellular data enabled (e.g., with *EDGE/3G/4G*), simply by sending push notification to the victim (a WhatsApp/Viber message) we can induce the device to wake up and initiate a Wi-Fi scan. We have noticed that the device typically initiates the scan within 5*s* from waking up. If cellular data is not enabled, we can also send an *SMS* to force the device to wake up from the sleep state.

We have observed that some devices, prior to going to sleep mode, scan for the neighboring APs with time periods that increase exponentially. Our initial brief analysis on iPhone 6s plus has shown another method to minimize the scan intervals for a device while being in sleep mode. By assuming that the device is connected to a known AP and continuously enabling iPhone to connect/force disconnect from the AP (e.g., by jamming a Wi-Fi channel), we can keep the device in a 60*s* scan period, as opposed to double or triple values otherwise.

2.3. Threat Model. As depicted in Figure 1, two different timelines are present during the attack. The configuration of the device timeline is unknown to the attacker, and the attacker timeline is limited in length (the attack can be performed only within the Wi-Fi vicinity of the victim). The attacker is constantly faking APs using different SSIDs. The goal is to test as many unique SSIDs within the available time, using as much as possible of the *match opportunity* time during the device scan period. If the attacker gets a connection attempt from the device, for a fake AP, the attack is considered a success.

Although the basic idea seems quite straightforward, the attack presents a lot of different challenges. How does the attacker know when the *match opportunity* occurs? What is the scanning and idle period length? How good is the Wi-Fi communication channel? Which SSIDs should be tested? How do we maximize the number of unique SSIDs we can test?

In the following section, we will answer these questions and describe the SSID Oracle attack.

3. Modeling the Problem

This section introduces a comprehensive mathematical model that depicts a general SSID attack. We begin with modeling the problem, after which we introduce the SSID Oracle attack accompanied by various attack variations and optimization techniques.

The proposed general model depicts a simple question: how can an adversary recover (at least one) SSID from the user's PNL and thus his previous whereabouts. The theoretical model comprises various SSID attacks scenarios, ranging from well-known passive sniffing to active attacks in which an adversary performs the attack with rogue access point or even advanced brute force attacks to user's PNL.

Before we pursue the detailed description of the attack, let us denote the following notation used to describe the location privacy *game*. First, we denote a targeted user's *preferred network list (PNL)* with $\mathscr{P}$, unknown by the adversary $\mathscr{A}$ prior to the attack. $\mathscr{D}$ denotes a dictionary list of (nontrivial) SSIDs prepared by $\mathscr{A}$ to be tested. $\mathscr{A}$ has a limited time period to execute the attack, so $\mathscr{D}$ needs to contain SSIDs which are highly likely to be present in $\mathscr{P}$. A confirmation that an SSID from $\mathscr{D}$ is successfully tested to be present in $\mathscr{P}$ is denoted with a *hit*. $\mathscr{A}$ has a goal of achieving a *hit*, under assumption that $\mathscr{P} \cap \mathscr{D} \neq \emptyset$ holds.

Definition 1. We define **success probability** $\mathbb{P}_{succ}^{L-priv}(\mathscr{A})$ as the probability that the adversary $\mathscr{A}$ performs a successful location privacy attack $L - priv$ and learns at least one SSID from $\mathscr{P}$:

$$\begin{aligned}\mathbb{P}_{succ}^{L-priv}(\mathscr{A}) &\triangleq \mathbb{P}[\mathscr{P} \cap \mathscr{D} \neq \emptyset, hit] \\ &= \mathbb{P}[\mathscr{P} \cap \mathscr{D} \neq \emptyset] \cdot \mathbb{P}[hit \mid \mathscr{P} \cap \mathscr{D} \neq \emptyset]\end{aligned} \tag{1}$$

As can be seen, the success probability $\mathbb{P}_{succ}^{L-priv}$ of the attacker $\mathscr{A}$ can be denoted as a product of two probabilities. Conditional probability $\mathbb{P}[hit \mid \mathscr{P} \cap \mathscr{D} \neq \emptyset]$ denotes the probability in which $\mathscr{A}$ observes a hit (a successful query), meaning that condition $\mathscr{P} \cap \mathscr{D} \neq \emptyset$ holds. Probability $\mathbb{P}[\mathscr{P} \cap \mathscr{D} \neq \emptyset]$ is responsible for building a good quality dictionary for the targeted victim such that the condition $\mathscr{P} \cap \mathscr{D} \neq \emptyset$ holds. Since the available opportunity window for $\mathscr{A}$ is time limited, potential attack's performances need to be optimized alongside having a quality dictionary.

There is a wide range of different strategies for revealing user's PNL (e.g., ranging from blind guessing to even physically reading $\mathscr{P}$ off the user's screen). Some devices in Active Service Discovery mode are transmitting the complete unencrypted PNL list to quickly establish a connection to a previously associated APs [17]. For $\mathscr{A}$ gathering unencrypted probe request packets represents a trivial problem for revealing $\mathscr{P}$ and will not be discussed any further in our work. Our model assumes that devices are periodically broadcasting empty probe requests or are passively scanning for APs. As we describe further in this paper, all Wi-Fi devices using active or passive service discovery are vulnerable to our attack.

3.1. Defining SSID Oracle. Recall, to verify that the victim's device holds an *SSID* within its preferred network list $\mathscr{P}$, that the adversary $\mathscr{A}$ can mount a rogue access point holding that specific SSID (i.e., by using the Karma tool [18].). If $\mathscr{A}$ confirms that the user initiated a connection, he concludes that the *SSID* is a part of the victim's PNL list ($SSID \in \mathscr{P}$).

Definition 2 (SSID Oracle). We denote victim as a binary response SSID Oracle $\mathscr{O}$. When $\mathscr{A}$ fakes an *SSID*, he is querying $\mathscr{O}$ for a binary outcome. If $SSID \in P$, then $\mathscr{O}$ returns a positive outcome; otherwise $\mathscr{O}$ does not respond. We denote the following notation:

$$\mathscr{O}(SSID) = \begin{cases} 1, & if\ SSID \in \mathscr{P} \\ 0, & if\ SSID \notin \mathscr{P} \end{cases} \tag{2}$$

In our attack $\mathscr{O}$ responds to queries only during the scanning period (Figure 1, *match opportunity*) and will

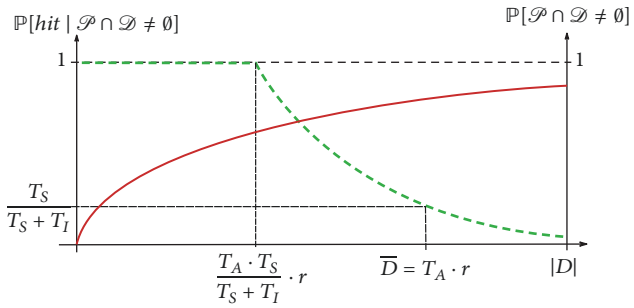

Figure 2: Conditional *hit* chance $\mathbb{P}[hit \mid \mathscr{P} \cap \mathscr{D} \neq \emptyset]$ and dictionary quality $\mathbb{P}[\mathscr{P} \cap \mathscr{D} \neq \emptyset]$ in comparison to the size of dictionary $|\mathscr{D}|$.

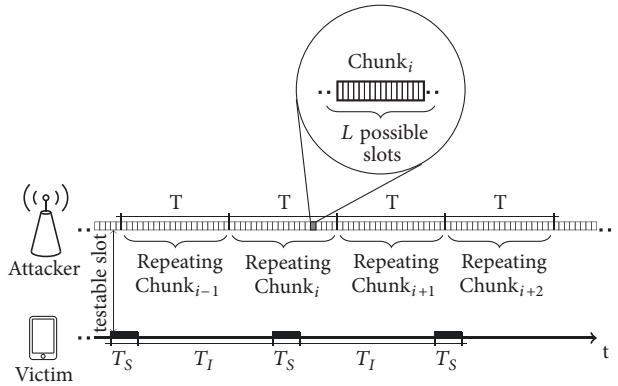

Figure 3: Attacker and victim timelines.

respond with a positive outcome only in the case that the queried SSID is within its $\mathscr{P}$. No response is considered to be a negative outcome. However, the success rate of querying $\mathscr{O}$ is subject to channel quality, which can potentially cause false negatives. In case of a low channel quality, the attacker cannot tell whether the cause for no response is $SSID \notin \mathscr{P}$ or the victim did not receive the query.

If $\mathscr{O}$ would respond to every query and the *match opportunity* would span during the entire attack time uninterrupted, our work would be much simpler, as we would only have to manage the problem of building a good quality SSID dictionary and test every SSID from that list one by one. A similar approach has already been covered by many papers describing dictionary or brute force attempts on password cracking [19, 26].

Since in our model the attacker does not actually know when that match opportunity period occurs (we assume the worst case scenario in our model with passive scan, as measured for iOS 10) and is subject to channel quality, it is necessary to create a model/algorithm that will allow the attacker to test every SSID at least once with as high probability as possible, within the match opportunity window.

3.2. SSID Oracle Attack. Independently of the type of scanning technique (Active Service Discovery with or without an empty probe request or Passive Service Discovery) in our model the Wi-Fi scan occurs periodically every T_I seconds (idle period in between scans) and lasts only for a short scanning period of T_S, such that $T_S \ll T_I$.

Figure 2 describes the general trade-offs that we face when executing SSID Oracle attack. As mentioned at the beginning of this section, to succeed in the location privacy game the adversary $\mathscr{A}$ would have to create a dictionary of (nontrivial) SSIDs that hold at least one SSID from $\mathscr{P}$. A simplest solution would be to increase the dictionary size so that the probability $\mathbb{P}[\mathscr{P} \cap \mathscr{D} \neq \emptyset]$ of finding at least one SSID approaches 1. This can be clearly seen from Figure 2, where the shape of the curve depends on different approaches to building $\mathscr{D}$. Our goal is to test SSIDs having a higher chance of being in $\mathscr{P}$ first, so $\mathbb{P}[\mathscr{P} \cap \mathscr{D} \neq \emptyset]$ will grow faster for small $\mathscr{D}$ with tendency of becoming linear for large $\mathscr{D}$. One such example approach on building dictionaries based on recommendation algorithm is described in Section 3.5. However, dictionary size depends on the opportunity window T_A, the time available for the attacker to execute the attack (Figure 1). If we take into account the fact that adversary $\mathscr{A}$ sends SSIDs (e.g., beacons) at maximum rate r, we can denote with *slot* a discrete time period in which one *SSID* from $\mathscr{D}$ is sent to the Oracle $\mathscr{O}$ (victim's device) within T_S period. Since T_S is the only valuable time period for $\mathscr{A}$ to conduct the attack, the maximum available number of time slots for the attacker within opportunity window T_A equals $(T_A \cdot T_S)/(T_S + T_I) \cdot r$. For this reason, the dictionary size tested on $\mathscr{O}$ satisfies the following condition: $|\mathscr{D}| \leq (T_A \cdot T_S)/(T_S + T_I) \cdot r$. As the dictionary size increases, the required time for the attacker $\mathscr{A}$ to test all SSIDs increases, so the conditional probability $\mathbb{P}[hit \mid \mathscr{P} \cap \mathscr{D} \neq \emptyset]$ drastically decreases.

Another observation shows just how big of an impact T_I has on the attack. If we assume that idle time is zero ($T_I = 0$), the victim's device would be constantly scanning the Wi-Fi channel. $\mathscr{A}$ can then create $\mathscr{D}$ such that the dictionary size equals the maximum number of time *slots* available in T_A ($\overline{\mathscr{D}} = T_A \cdot r$). $\mathscr{A}$ could easily test many more SSIDs from the dictionary against the victim's device within T_A (by taking into account the fact that all queries arrive to the Oracle $\mathscr{O}$ (there are no collisions in Wi-Fi channel) and that there are no retransmissions.).

In our model, we do not know when T_S or T_I occurs, just a measurement of their lengths. To be able to execute the attack, despite not knowing when a new scanning period starts, a logical solution (intuition) would be to create small *chunks* of size L testable slots, fill them with SSIDs from $\mathscr{D}$, and retransmit chunks for the duration of time period T until we are convinced that $\mathscr{O}$ received all L queries from the chunk at least once. After the chunk retransmission time T, $\mathscr{A}$ sends the following chunk containing another L slots. The proposed method is described in detail in Figure 3.

The attacker strategy s can be described by using the following triplet $\mathscr{D}$, L, and T:

$$s = (\mathscr{D}, L, T) \tag{3}$$

Our goal is to find the optimal strategy s^* which will result, given the available time period T_A (opportunity window), in the best SSID Oracle attack execution:

$$s^* = \arg\max_{(\mathscr{D},L,T)} \mathbb{P}[hit \mid \mathscr{P} \cap \mathscr{D} \neq \emptyset] \cdot \mathbb{P}[\mathscr{P} \cap \mathscr{D} \neq \emptyset] \tag{4}$$

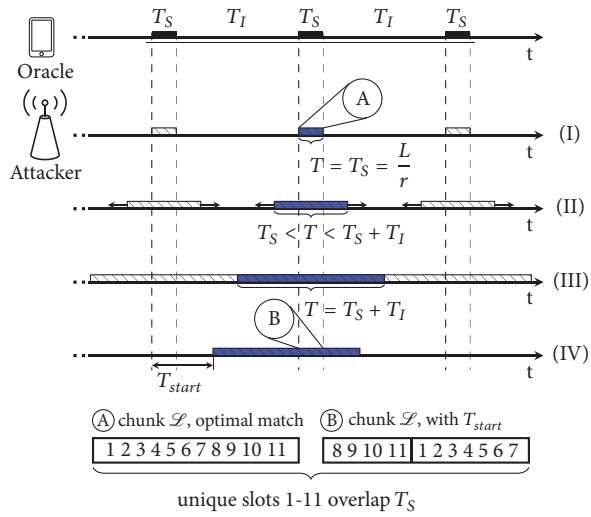

Figure 4: Choosing optimal chunk transmission time T.

The following sections focus on defining s^*. We start by discussing and defining L and T parameters which affect the efficiency of attack execution. Later in Section 3.5 an example method for defining and building a good quality $\mathscr{D}$ is presented.

3.3. Finding Optimal Parameters L^* and T^*. As we mentioned in the previous section, chances of making Oracle $\mathscr{O}$ respond to a query with a positive answer depend on multiple parameters, out of which some are predetermined by the Oracle (smartphone) implementation (i.e., periods T_I, T_S), while some are controlled by the adversary $\mathscr{A}$ (i.e., dictionaries $\mathscr{D}$, L, and T). In this section, our goal is to find the optimal parameters for the chunk size L^* and chunk retransmission time period T^*. Recall from Figure 3 that it is necessary to find a strategy that guarantees (with high probability) that every *slot* from the chunk of size L hits the scanning interval T_S.

Let us consider the scenario in which the attacker $\mathscr{A}$ is omniscient; i.e., $\mathscr{A}$ knows the exact moment at which $\mathscr{O}$ initiates the search for neighboring SSIDs. Since $\mathscr{A}$ knows when scan initiates and also its duration T_S, he can adapt the attack transmission start time T_{start} towards $\mathscr{O}$ at maximum rate r, as can be seen in Figure 4-I. Now we can easily conclude that the maximum number of unique slots that can fit into T_S equals to $r \cdot T_S$ (recall that r denotes a rate at which chunks are transmitted). Due to the fact that during the period T_I the Oracle $\mathscr{O}$ cannot receive any query from $\mathscr{A}$, we can denote $r \cdot T_S$ as a single chunk $\mathscr{L}_i$, i.e., $|\mathscr{L}_i| = L_i = r \cdot T_S$. Now, referring back to our attacking model (Section 2), $\mathscr{A}$ is not omniscient and cannot actually know when T_S starts; therefore it is necessary to find such a strategy that the complete $\mathscr{L}$ finds T_S period.

Please observe from Figure 4 that since the $T_S + T_I$ intervals are periodically repeating, the easiest solution for $\mathscr{A}$ would be to retransmit chunks $\mathscr{L}_i$ towards left and right (Figure 4-II) and expand them until it reaches the expansion of chunk $\mathscr{L}_{i-1}$ on the left and chunk $\mathscr{L}_{i+1}$ on the right (Figure 4-III). How does this help the attacker? Figure 4-IV shows that since chunk retransmission period equals $T = T_S + T_I$ and $L = r \cdot T_S$, regardless of when T_{start} period actually occurs, the complete chunk is still going to hit the appropriate scanning period T_S, with $T_{start} \in [0, T_S + T_I]$. The zoomed sections in Figure 4, A (in case of all-knowing $\mathscr{O}$ scenario) and B (in case of any other scenario), show that the same unique SSIDs contained in $\mathscr{L}$ chunk will overlap the scanning period T_S in both cases.

For this reason the optimal chunk size equals the number of slots that can fit in the scanning period T_S:

$$L^* = r \cdot T_S, \tag{5}$$

whereas the optimal time interval to retransmit $i - th$ chunk equals the sum of T_I and T_S:

$$T^* = T_S + T_I. \tag{6}$$

Given the attacker's opportunity window T_A, we can calculate the maximum number of unique testable slots N_{slot} for our strategy s^*:

$$N_{slot} = \frac{T_A}{T_S + T_I} \cdot L, \tag{7}$$

We have shown that by using optimal T^* we can achieve the same number of slots N_{slot} compared to the omniscient $\mathscr{A}$ we started this discussion from, although the actual attacker $\mathscr{A}$ cannot tell the exact moment at which $\mathscr{O}$ initiates T_S.

The following conclusion sums up our observations so far: when defining s^*, execution of SSID Oracle attack with parameters L^* (5) and T^* (6) will **maximize the available number of unique slots** N_{max}.

However, in low channel quality conditions there is always a chance that query sent from $\mathscr{A}$ does not get to $\mathscr{O}$. If we have a hit ($\mathscr{O}$ responds to a query with positive outcome), there is a chance that the response does not get to $\mathscr{A}$, so we are introducing a new parameter-probability p that SSID Oracle successfully receives and responds to a query.

For this reason it may not be optimal to fill all the slots with unique SSIDs; thus the size of the dictionary might have to decrease even further.

$$\overline{\mathscr{D}}_{max} \leq N_{slot}. \tag{8}$$

3.4. Matching Dictionary to Slots. In previous sections we carried out the analysis of proposed model in good quality channel environments ($p \approx 1$). We have shown that the optimal chunk retransmission time is the period $T^* = T_S + T_I$, while the chunk size equals the number of queries that can fit the scan interval $|\mathscr{L}| = L^* = r \cdot T_S$. In this scenario, $|\mathscr{D}|$ was equal to the maximum number of testable slots $|\mathscr{D}| = N_{slot}$ within T_A.

However, what can be done in scenarios of low quality channel $p < 1$? In such scenarios, not all slots that attacker $\mathscr{A}$ sends within scan period T_S will be received by $\mathscr{O}$, the same way the responses sent by $\mathscr{O}$ (in case we have a hit) will not be received by $\mathscr{A}$ (all these parameters depend on attacker's Wi-Fi card quality, victim's Wi-Fi card quality, channel quality,

distance from the attacker and the victim, etc.). To be more precise, in low quality channel conditions it could be better to retransmit SSIDs that have a larger probability of occurrence in the victim's PNL list, rather than to transmit N_{slot} different SSIDs for a given T_A.

Let us use $\mathscr{D}'$ to denote the set of all SSIDs known by $\mathscr{A}$, whereas not all *SSID*s from $\mathscr{D}'$ have the same probability of being part of the victim's PNL list $\mathscr{P}$. Our goal is to maximize $\mathbb{P}_{succ}^{L-priv}$ thus finding the optimal $\mathscr{D} \subset \mathscr{D}'$ to be tested within window of opportunity T_A. The event of a successful $\mathcal{O}$ test, with a positive result, is denoted by $h_i, \forall i \in \mathscr{D}'$. To find the best $\mathscr{D}$, $\mathscr{A}$ has to use different tools (such as the example recommender system found in Section 3.5) to assign a chance that an SSID contained in $\mathscr{D}'$ is also contained in the victim's PNL. For this purpose, we are introducing a new subjective probability parameter p_i assigned to every SSID in $\mathscr{D}'$ and calculated by $\mathscr{A}$.

$$p_i \in [0, 1] \quad \forall i \in \mathscr{D}' \tag{9}$$

Recall that we denote with *slot* a discrete time period in which one *SSID* is sent to the victim's device within T_S period. It is our goal to determine the number of slots every SSID in $\mathscr{D}'$ should take:

$$n_i \in [0, N_{slot}] \quad \forall i \in \mathscr{D}' \tag{10}$$

In order to characterize n_i, we proceed as follows:

$$\begin{aligned} \mathbb{P}_{succ}^{L-priv} &= \mathbb{P}\left[\mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \cdot \mathbb{P}\left[h \mid \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \\ &= \mathbb{P}\left[\mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \\ &\quad \cdot \mathbb{P}\left[h_1 \vee \cdots \vee h_{|\mathscr{D}'|} \mid \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \end{aligned} \tag{11}$$

The dictionary quality equation part of (11) resolves to

$$\begin{aligned} \mathbb{P}\left[\mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] &\triangleq \sum_{i \in D'} \mathbb{P}\left[i \in \mathscr{P} \mid \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \\ &= \sum_{i \in D'} p_i \leq 1, \end{aligned} \tag{12}$$

Attack execution equation part of (11) resolves to

$$\begin{aligned} \mathbb{P}\left[\vee_i h_i \mid \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] &\leq \sum_{i \in \mathscr{D}'} \mathbb{P}\left[h_i \mid \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \\ &\overset{(1)}{=} \sum_{i \in \mathscr{D}'} \mathbb{P}\left[h_i, i \in \mathscr{P} \mid \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \\ &= \sum_{i \in \mathscr{D}'} \Big(\mathbb{P}\left[h_i \mid i \in \mathscr{P}, \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right] \\ &\quad \cdot \mathbb{P}\left[i \in \mathscr{P} \mid \mathscr{P} \cap \mathscr{D}' \neq \emptyset\right]\Big) \overset{(2)}{=} \sum_{i \in \mathscr{D}'} \left(1 - (1-p)^{n_i}\right) \\ &\quad \cdot p_i \end{aligned} \tag{13}$$

where (1) follows from $\mathbb{P}[A \mid B] = \mathbb{P}[A, C \mid B] + \mathbb{P}[A, \neg C \mid B]$, whereas latter part of the equation in our case is equal to 0 ($i \notin \mathscr{P}$) and (2) comes the under assumption that Oracle queries are mutually independent of each other (sum of probabilities). Please note that due to burst effects in Wi-Fi channel [27] we may not always face such scenario, since there is a chance that consecutive Wi-Fi packets are more likely to fail to transmit if previous packets also failed. Given the premises $N_{slot} \gg L$ and $T \gg T_S$ we can see that different L slots are being tested couple of seconds apart, meaning that different chunks are not subject to the same error burst. In case of retransmitting SSIDs, we use interleaving to tackle the error burstiness.

We are now ready to set the optimization problem, whose solution will give $\mathscr{A}$ the required parameters n_i. Please note that the SSID Oracle attack requires $n_i \in \mathbb{N}_0$, but we are using linear optimization to solve the optimization problem. To execute an attack, it is required to round n_i calculated from our model, as will be shown in simulations later.

$$\begin{aligned} \max_{n_1, \ldots, n_{\mathscr{D}'}} \quad & \sum_{i \in \mathscr{D}'} \left(p_i - p_i (1-p)^{n_i}\right) \\ \text{s.t.} \quad & \sum_i n_i \leq \frac{T_A T_S}{T_S + T_I} \cdot r = N_{slot} \\ & n_i \geq 0 \\ & p_i \in [0, 1] \\ & \sum_i p_i \leq 1 \\ & p \in [0, 1] \end{aligned} \tag{14}$$

We approach the optimization problem (14) using Lagrange multipliers method. Objective function has unique solutions based on its concave nature.

Lemma 3. *The relation between SSID subjective correctness probability p_i, channel quality p, and number of testing attempts n_i for every SSID is*

$$n_i^* - n_j^* = \frac{\log\left(p_j / p_i\right)}{\log(1-p)} \tag{15}$$

As can be seen from (15) and Figure 5, for good channel ($p > 0.95$), the attacker does not significantly increase his hit chance by retransmitting SSIDs with higher probability of occurrence. On the other hand, a bad channel ($p < 0.5$) brings up the importance of having a good SSID dictionary generator algorithm.

Theorem 4. *SSID Oracle attack execution strategy s^* should be achieved by picking $L^* = r \cdot T_S$ and $T^* = T_S + T_I$ parameters. The optimal execution strategy s^* maximizes the number of unique testable slots N_{slot}. When matching $\mathscr{D}$ to N_{slot}, the condition $n_i - n_j = \log(p_j/p_i)/\log(1-p)$, $\forall i, j \in \mathscr{D}'$ must be fulfilled to achieve the best attack performance in case of a poor quality channel p.*

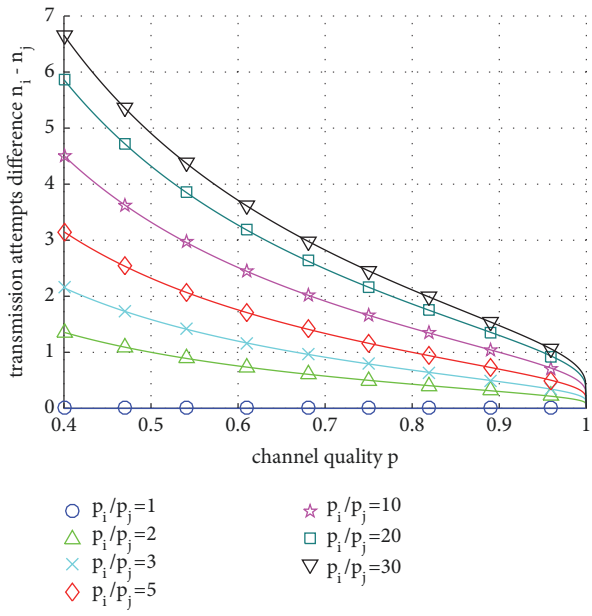

FIGURE 5: Transmission attempts difference $n_i - n_j$ as a function of channel and SSID quality (based on Lemma 3).

In the next subsection, we will introduce an example dictionary creation method, based on top-N recommendation algorithm, which can provide us with the required SSID importance parameters p_i.

3.5. Building a Dictionary. In this subsection we introduce an example dictionary generation algorithm, based on a recommendation algorithm. We will only briefly cover the approach and later provide the experimental results in Section 4.3. Please note that there can be multiple different dictionary creation approaches, depending on the information the attacker has on the potential victim. One goal could be to gather as many pairs of Wi-Fi MAC address and SSID to do statistical analysis of a crowd (similar work to many papers mentioned in our related work). In that case, having a more general dictionary is the best approach. Another goal could be the deanonymization of a specific person or a group by linking the victim to his MAC address. In that case, the dictionary should be prepared for that specific scenario, containing related SSIDs from victims neighborhood. Nevertheless, our SSID Oracle attack strategy will maximize the potential successful outcome for any provided dictionary.

In order for the attack to be successful, the dictionary has to be

(i) compact: the size of the dictionary $\mathscr{D}$ should be small enough so that performing the attack is feasible in reasonable time

(ii) precise: the dictionary should reflect victim's PNL as much as possible (contain as many SSIDs that are in the user's PNL as possible)

The method we used for building a dictionary for the attack is based on collaborative filtering, namely, a modified version of Item-based top-N recommendation algorithm as seen in [21, 28]. It outputs a list of N recommended items that user might prefer/consume/buy/visit based on the current knowledge of user's preferences/history and the knowledge of preferences/history of other users (training dataset). Cosine based similarity has shown to produce the best results.

After building a list of N recommended SSIDs for each test user, it is compared with the original PNL and the hit-rate (HR) is calculated:

$$HR = \frac{\#\text{ of SSIDs recommended that are in users PNLs}}{\#\text{ of SSIDs in users PNLs}} \quad (16)$$

An HR value of 1.0 indicates that the algorithm was always able to recommend the hidden item, whereas an HR value of 0.0 indicates that the algorithm was not able to recommend any of the hidden items [21].

Our algorithm uses two parameters which affect the effectiveness of the modified recommender system. The number of similar SSIDs we want to store for each SSID in the learning process is denoted by $k \in \mathbb{N}$. The growth of the exponential equation used for calculating the importance of SSID repetition in the training set data is controlled by parameter $A \in \mathbb{R}$.

In the next section we will perform various SSID Oracle attacks using our setup.

4. Practical Tests and Simulations

In this section we present the performed practical tests, simulations, and an example dictionary creation algorithm based on top-N recommendation algorithm. The achieved results have been compared with the appropriate parts of our model.

4.1. Experimental Analysis of Chunk Size L and Retransmission Time T. Extensive tests have been carried out to show the effectiveness of SSID Oracle attack and correctness of chunk size L^* and retransmission time period T^* optimal parameters on tests with real devices and in real-world scenarios. To be more precise, our tests were based on a modified *Airbase-ng* tool on Ubuntu machine equipped with D-Link DWA-556 Wireless N PCI-E Desktop Adapter placed in monitoring mode used to "... encourage clients to associate with the fake AP"[29]. We slightly modified Airbase-ng in a way that sends beacon packets from a predefined chunk of fixed size L, while it does not reply to any authentication request packets (nor probe request packets). Beacon packets were sent only on a single Wi-Fi channel (Wi-Fi channel 1 on 2.4 *GHz* in our scenario). To accomplish focusing on a specific Wi-Fi channel, in parallel with Airbase-ng we used Airodump-ng tool. On the other hand, to capture authentication requests, we also ran *tshark*, filtering out sought authentication request packets. Every test chunk $\mathscr{L}$ holds one SSID from $\mathscr{P}$, i.e., $|\mathscr{L} \cap \mathscr{P}| = 1$. For different chunk sizes $\mathscr{L}$ ($L \in \{50, 60, 80, 100\}$) performed in our tests we transmit beacon packets from chunk $\mathscr{L}$ for T consecutive seconds ($T \in \{2, 3, 4, \ldots, 10\}$). Interestingly, a similar test with multiple APs was implemented using Airbase-ng tool in [19], but the authors had

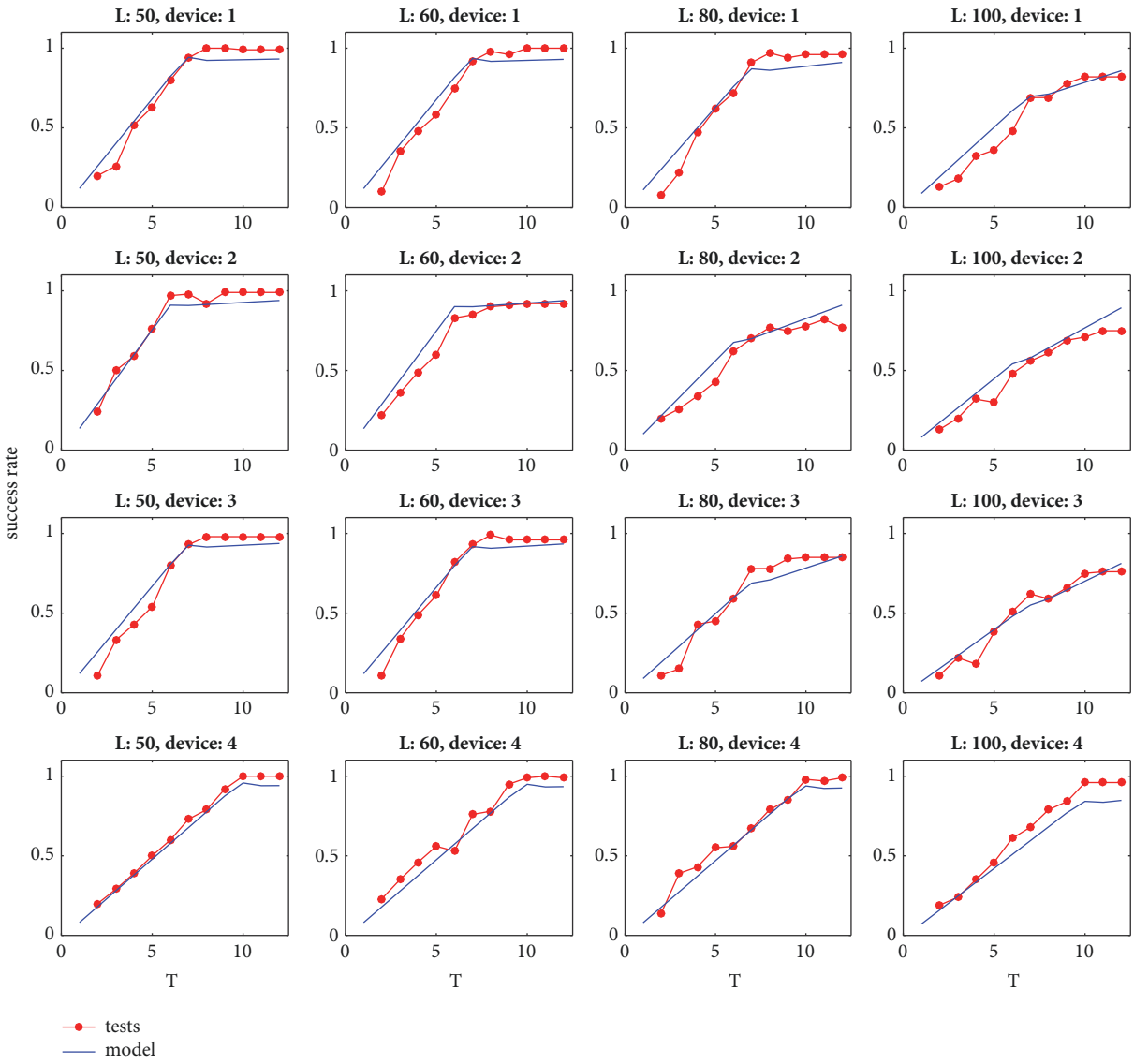

FIGURE 6: Model comparison with test results for different devices and chunk sizes, using $r = 500(SSIDs/sec)$.

problems with handling large number of SSIDs, so they limited their attack to a smaller number of SSIDs, i.e., 5 SSIDs, which is significantly smaller than our SSID Oracle attack with chunk sizes up to 100 APs (and even more). We repeated this test 100 times for each fixed period T and chunk size L. It is also important to note that since T_{start} can appear uniformly at random within $T_I + T_S$ (Figure 3), a random delay between two consecutive tests was induced. To enable shorter periodic scanning behavior (periods $T_S + T_I$), smartphone's screen is powered ON and the default Wi-Fi finder program is opened in every test. This was done solely for practical reasons since running experiments for higher scanning intervals, e.g., "Display off" as mentioned in Section 2.1, would take too long (Figure 7 experiments alone had been running for a month).

Figure 6 shows the test results were carried out at Samsung Galaxy 3, Sony Xperia X8, LG P350, and Samsung S5 Mini devices. The dotted lines present success rate of response reception to query for various parameters of chunk retransmission period T and a fixed chunk size L. It is interesting to observe that for small transmission periods T there is a small chance that the Oracle responds with a success (a *hit*). Since idle period T_I is larger than T and given that the start of transmission period T_{start} starts uniformly at random within $[0, \ldots, T_S + T_I]$, for small T there is a high probability that attacker transmits a query within idle period. By increasing the chunk retransmission period T the success rate increases rapidly up to a point $T_S + T_I$. Note that this period is not equal for all devices; i.e., for Samsung Galaxy 3 it was $7s$ and for Sony Xperia X8 it was approximately $6s$, while LG P350 was $7s$. Indeed, in an additional study the time period between two consecutive probe packets corresponded to these intervals (in Active Service Discovery, when devices initiate the scan for neighboring APs on a specific Wi-Fi channel, they send a burst of probe request packets [30]). Moreover, it is important to note that chunks $\mathscr{L}$ were sent on a

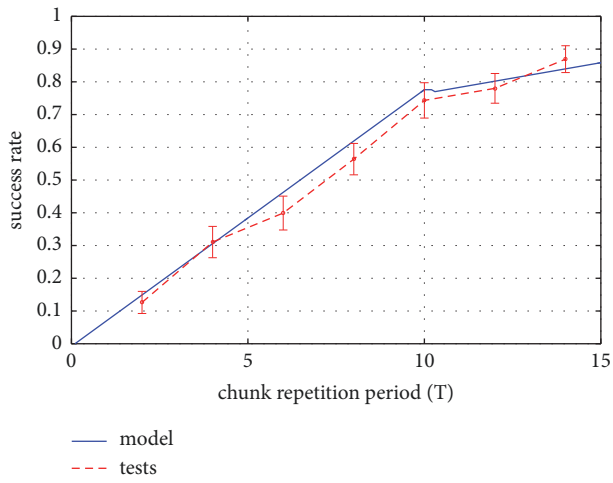

Figure 7: Comparison of our model with experimental results on Sony Xperia miro for $L = 60$.

Table 1: Scanning and idle intervals for test devices.

Device	T_S	$T_I + T_S$	p
1. Samsung Galaxy 3	150 ms	7 s	0.97
2. Sony Xperia X8	120 ms	6.2 s	0.95
3. LG	120 ms	7 s	0.95
4. Samsung S5 Mini	180 ms	10 s	0.95
5. Sony Xperia Miro	100 ms	10.1 s	0.95

single Wi-Fi channel (Wi-Fi channel 1 in our case). Therefore, the effective scanning interval T_S corresponds to the scanning interval of a single channel (if we eliminate the possibility of capturing communication on non-overlapping channels), which approximates to $0.1 - 0.12s$. Interestingly, we can see in Figure 6 that the best results for success rate were indeed achieved for chunks sizes of $L \leq 60$. Please note that since in our tests every beacon packet was sent approximately every 2 ms ($r \approx 500s^{-1}$), sending the complete chunk L will take the exact amount of time that corresponds to the scan interval of $T_S \approx 0.12s$ of a smartphone device ($L/r \approx 0.12s$). We can also observe the impact of imperfect Wi-Fi channel where Oracle does not reply to all queries successfully, although $T_S = L/r$ and $T = T_S + T_I$; i.e., the success rate is not 1 but instead approximates to $p \approx 0.98$. In Table 1 we give detailed results for periods T_S, $T_I + T_S$, and the probability p for every tested device.

To verify the correctness of experimental results we also developed a mathematical model depicting the hitting probability $\mathbb{P}[hit]$ for SSID Oracle attack—the probability that the attacker $\mathscr{A}$ observes a *hit*, given that the observed chunk holds at least one *SSID* from $\mathscr{P}$ ($\mathscr{L}_i \cap \mathscr{P} \neq \emptyset$). We evaluate this probability as a function of parameters T and L and rate r controlled by the $\mathscr{A}$, and parameters T_S and T_I given by Oracle's specification, as well as the channel quality p. Our model also assumes that the chunk transmission time T_{start} starts uniformly at random within the period $[0, \ldots, T_S + T_I]$ and that the sought *SSID* is also placed uniformly at random within the observed chunk $\mathscr{L}$.

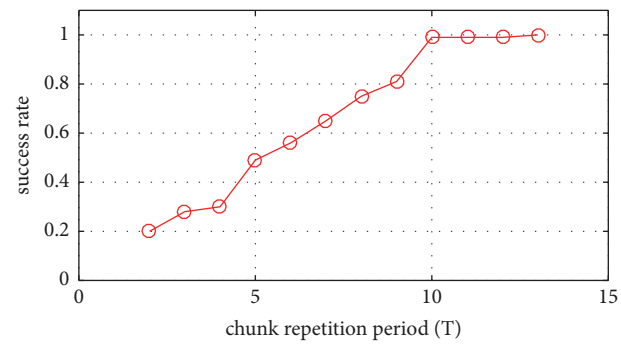

Figure 8: Achieved results for channel quality $p \approx 1$.

In our analysis (see Appendix A) we obtain the expression for probability $\mathbb{P}[hit]$ which is solved numerically.

The experimental results, along with the numerical ones, are shown in Figure 6. As we can see from the figure, the model quite accurately predicts the probability for parameters given in Table 1. We also carried out extensive tests with Sony Xperia miro device and presented them with 95% confidence interval. For every chunk transmission period T we carried out 10000 tests, while the chunk size was $|\mathscr{L}| = 60$. The experiments give us the values for $T_S = 100ms$, while $T_S + T_I \approx 10.01s$. By plugging these values into our model (Appendix) we can see that our model quite accurately predicts the probability $\mathbb{P}[hit]$ (Figure 7).

As mentioned before, our experiments were conducted in an area where other Wi-Fi devices were also transmitting, so there has been some interference present. To try to get almost perfect Wi-Fi conditions ($p = 1$) we did another test in a controlled and clean environment with no other devices transmitting in Wi-Fi band. The success rate of almost a 100% has been achieved on the device for $T_I + T_S = 10s$, as can be seen from Figure 8.

4.2. Retransmission Simulations for Low Quality Channel Conditions ($p < 1$). In order to show the effectiveness of SSID Oracle attack in low quality channel conditions ($p < 1$), we implemented a simulator in MATLAB. The simulator gives us a good understanding about SSID retransmissions (testings) under low quality channel conditions, thus allowing us to manipulate various parameters, from the quality of the dictionary (the probability of every *SSID* being part of victim's PNL list), up to the number of testings n_i of a single SSID (SSID retransmissions in different chunks). For the purpose of our simulator we used the following variables: $T_A = 1212s$, $T_I = 10s$, $T_S = 100ms$, $L = 50$, and $r = 500^{-1}s$. Every point in the results was simulated 200 000 times.

Example 5. In this example we verify the effect of retransmitting (testing) SSIDs multiple times. By retesting an SSID multiple times, i–th SSID from $\mathscr{D}$ fits more than one slot (has more than one chance of a *hit*) within the Oracle's T_A period. Since the number of available N_{slot} is limited in T_A, this also means that testing one SSID more than once results in not having available slots intended to test all SSIDs in $\mathscr{D}$. Simulator test dictionary $\mathscr{D}$ contains SSIDs

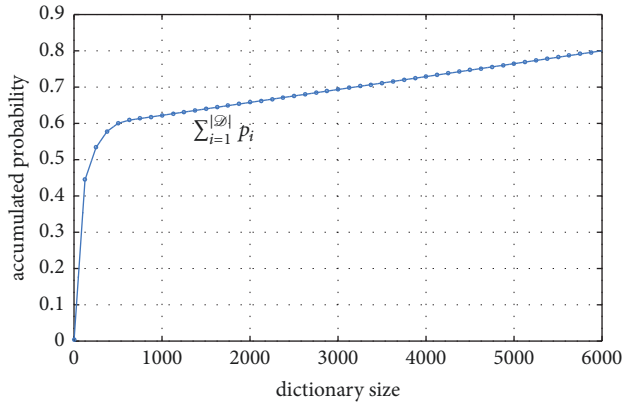

Figure 9: Cumulative probability distribution in test dictionary.

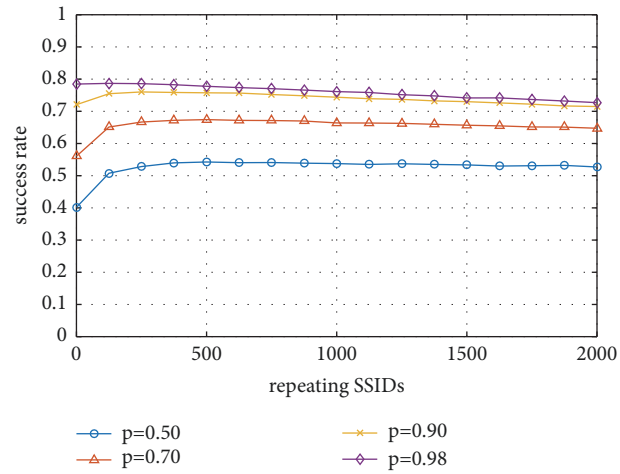

Figure 10: Success rate for different channel quality p when swapping SSIDs in test dictionary.

assigned with corresponding probabilities contained within user's PNL list $p_i = \mathbb{P}[i \in \mathscr{P} \mid \mathscr{P} \cap \mathscr{D} \neq \emptyset]$, $\forall i \in \mathscr{D}$ (normalized to $[0, 1]$). Please note that SSIDs in $\mathscr{D}$ are ordered according to the descending probabilities ($p_i \geq p_{i+1}, \forall i$), such that SSID with higher probability will be tested first. In our example, dictionary contains 6000 SSIDs. Figure 9 shows the cumulative probability distribution of SSIDs within $\mathscr{D}$. We can observe that $\sum_{i=1}^{1000} p_i = 0.62$, whereas if we observe the complete dictionary of 6000 SSID we will have $\sum_{i=1}^{6000} p_i = 0.8$. Therefore, with a quality dictionary, in scenarios with low channel quality, an attacker $\mathscr{A}$ will test those SSIDs having a higher probability of occurrence in $\mathscr{P}$ at the cost of not testing SSIDs having a lower probability of occurrence.

Figure 10 presents the achieved test success rate for various channel qualities p. The available transmission time $T_A = 1212s$ allows us to attempt tests for 6000 slots ($N_{slot} = 6000$). Simulation will test the first N_{repeat} SSIDs twice ($n_i = 2, \forall i \in \{1, \ldots, |\mathscr{D}| - N_{repeat}\}$, s.t. $p_i \geq p_{i+1}$), which have a higher probability of occurrence in $\mathscr{P}$, at the cost of not testing the least significant N_{repeat} SSIDs. The goal is to

Table 2: Achieved increases in Simulation 1.

p	N_{repeat}	succ. rate	increase to $N_{repeat} = 0$
0.5	500	0.5426	35.07%
0.7	500	0.6742	20.29%
0.9	250	0.7602	5.49%
0.98	125	0.7868	2.8%

Table 3: Optimal n values for Simulation 2.

p	n_{1-125}^{opt} test	n_{1-125}^{opt} model	increase to $n_{1-125} = 1$
0.5	5	4.9069	32.15%
0.7	3	2.8250	12.9%
0.9	2	1.4771	2.63%
0.98	1	0.8694	0%

observe how this exchange affects the overall hit success rate for different channel qualities. The following holds:

$$n_i = \begin{cases} 2, & \forall i \in \left[1, N_{repeat}\right] \\ 0, & \forall i \in \left[|\mathscr{D}| - N_{repeat}, |\mathscr{D}|\right] \\ 1, & \forall i \in \left(N_{repeat}, |\mathscr{D}| - N_{repeat}\right) \end{cases} \quad (17)$$

The importance of finding a method/algorithm for building a quality dictionary for targeted users was pointed out in Figure 10 and Table 2, especially in the scenarios with low channel quality. More precisely, by reducing the dictionary by $N_{repeat} = 500$ and giving the opportunity to (re)test first N_{repeat} SSIDs twice ($n_i = 2$), we achieve an overall increase in probability of hit of up to 35.07% and 20.29% for channel qualities $p = 0.5$ and $p = 0.7$, respectively. On the other hand, by incrementing N_{repeat} for good quality channels, the overall success rate decreases, since the actual dictionary to be tested now decreases to the size of $|\mathscr{D}| - N_{repeat}$, thus not giving an opportunity to test SSIDs with lower probability within the initial dictionary $\mathscr{D}$.

Example 6. In the following example we verify, both through simulations and theoretically, the optimal number of slots assigned to the first N_{repeat} SSIDs in scenarios with different channel quality p that would maximize the probability of hit. We use the following properties for our model: $|\mathscr{D}| = 6000$, $N_{repeat} = 125$, $\sum p_i = 0.8$, and $p_i \backslash p_j = 30$, $\forall i \in \{1, 2, \ldots, 125\}$, $\forall j \in \{\mathscr{D} \setminus i\}$. Simulation is performed for $n_i \in \{1, \ldots, 11\}$ values.

The results can be found in Figure 11 and Table 3. Indeed, from the results we can see that the estimated maximum number of slots assigned to the first N_{repeat} SSIDs corresponds to the ones obtained through simulations. We also show that it pays off for $\mathscr{A}$ to re-test SSIDs with high probability of occurrence at the expense of not testing the ones with low probability of occurrence. As expected, the highest increase in the hit probability will be obtained during the lowest channel quality p.

In the next section, we approach the problem of building a good quality dictionary for the targeted user, i.e., a dictionary

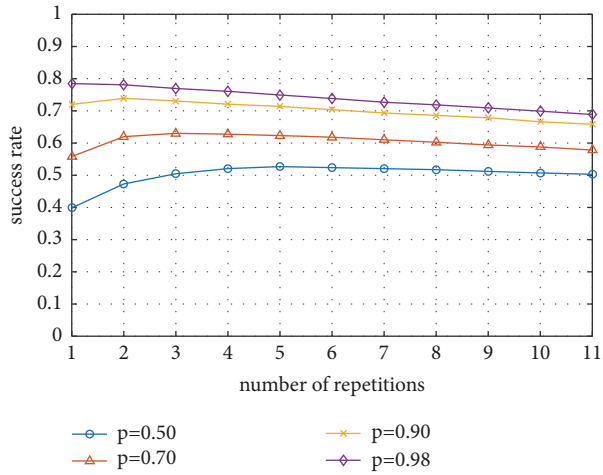

Figure 11: Success rate for different channel quality p and SSID repetitions n.

in which a probability of occurrence of the first N_{repeat} SSIDs within victim's PNL list will be high.

4.3. Dictionary Creation Results. In this section we will present the achieved results based on the top-N recommender system algorithm we discussed in Section 3.5. For testing the performance of the system we collected users' PNLs from Apple devices, mainly from the visitors of a big music festival in our country. 4426 users (different MAC addresses) were collected and 15095 SSIDs, with total of 23701 MAC-SSID pairs (average PNL size of 5.35). Another interesting starting dataset is also provided by the authors of [31] where SSIDs were gathered on multiple locations in Rome, Italy.

To get more accurate and statistically valid results, instead of doing one test experiment, a 10-fold cross-validation method is used where the dataset is divided into 10 partitions and in each of 10 folds (reruns of the experiment) one partition is used as test set and the rest as training set. The final hit-rate (HR) is calculated as the average of hit-rates for all folds.

It is also possible to optimize the performance of the system by running cross-validation with different parameters of the system and pick the set of parameters which produce the best result [32]. Different similarity functions are also tested (cosine based or conditional probability based). The results are shown in Table 4.

We see that the best results can be obtained by setting $A = 100$ and $k = 20$, even though the system is quite stable on parameters variations. To show how well the system performs on our dataset in comparison with other (benchmark) datasets, we have compared the results of our experiments with the ones done in [21] (which are modified to fit the classical recommender scenario), and showed the results in Table 5.

Clearly the performance is comparable with datasets from the classical scenario of recommender system usage (despite the fact that the density of our dataset is quite low), which justifies its application here as well. There are other possibilities of making and improving dictionary which can be taken into consideration in future work, like including human knowledge, categorizing people by their preferences/life habits, or applying additional rules for specific scenarios.

5. Related Work

A lot of work covers the location privacy problem for devices disclosing their PNL. Numerous papers focus on the IEEE 802.11 connection scanning and initialization protocols: AS, AS with broadcast, or PS [16, 24, 33, 34]. Those papers are relevant for our research as it covers scanning and idle times and showcases the tendency for the periodical Wi-Fi scans. LAPWiN proposes a location based protection mechanism to protect one from privacy leaks [30]. Such protection mechanism would directly mitigate the kind of privacy attack we are attempting; however that standard has not been implemented by the Wi-Fi card manufacturers yet. The authors in [23] are using Wi-Fi behavior to do aerial search and rescue operations. The authors are monitoring probe request packets using equipment mounted on drones in order to detect the location of a user during the search and rescue operation.

Others focus on making conclusions about a user from their PNL. Signals from the crowd [7] uses gathered SSIDs to discover user's country of origin, device manufacturer, and other, as is [8, 9, 35]. SSIDs in the Wild [36] are mapping SSIDs to real-world locations, and [10] finds social relation between users by matching PNL. WiFiPi [11] tracks user movement at mass events using a combination of MAC address and SSIDs, using a deployed sensor network.

Linking a MAC address gathered from a Wi-Fi packet to an actual person (MAC de-anonymization) appears to be challenging. Reference [26] uses beacon reply attack and fakes user's known SSIDs to trigger his phone to connect, thus doing a MAC address matching. Beam me up, Scotty [37] has an interesting approach to Wi-Fi assisted geo location where they fake an AP from another location causing services like Twitter to display the fake location as the origin of a tweet. The authors in [38, 39] showcase an entire network of sensors doing location privacy attack in the city of London gathering data on users movements and whereabouts.

To complicate tracking and privacy leaks, user equipment manufacturers have started using MAC address randomization. However, the authors in [19, 40, 41] showed that randomizing MAC address does not increase privacy, as the devices are sending probe requests that contain APs from the PNL, proving that privacy protection is indeed a considerate issue for the manufacturers.

The related work mentioned in this section so far focuses on passive monitoring of probe requests and various conclusions one can deduce from gathered PNLs. The devices using active scanning with broadcast packets or passive scanning are not vulnerable to passive monitoring, so an active attack is needed.

The attempt at actively faking an AP and thus revealing user's PNL has not been researched in detail, to the best of our knowledge. Active attacks on user's PNL have been mentioned in some previous work [19, 26] where the authors are mounting fake APs containing user's known SSID in order

Table 4: Cross-validation hit-rate for different parameters for $N = 1000$.

	Cosine based				Conditional probability based			
	$k = 10$	$k = 20$	$k = 40$	$k = 60$	$k = 10$	$k = 20$	$k = 40$	$k = 60$
$A = 5$	32.69%	32.43%	32.40%	32.37%	32.74%	32.50%	32.55%	32.57%
$A = 10$	32.65%	32.47%	32.41%	32.34%	32.68%	32.74%	32.77%	32.64%
$A = 20$	32.62%	32.43%	32.37%	32.37%	32.75%	32.80%	32.73%	32.84%
$A = 100$	32.75%	32.72%	32.69%	32.67%	32.73%	**32.89%**	32.86%	32.76%

Table 5: Performance comparison with other datasets.

Dataset	n	m	# of records	Density	Average # items/user	HR for experiment
ctlg1	58565	502	209715	0.71%	3.58	41.5%
ctlg2	23480	55879	1924122	0.15%	81.95	15.4%
ctlg3	58565	39080	453219	0.02%	7.74	54.0%
ccard	42629	68793	398619	0.01%	9.35	17.6%
ecmrc	6667	17491	91222	0.08%	13.68	17.4%
em	8002	1648	769311	5.83%	96.14	40.5%
ml	943	1682	100000	6.31%	106.04	27.2%
skill	4374	2125	82612	0.89%	18.89	37.3%
ssids	**4426**	**15095**	**23701**	**0.04%**	**5.35**	**32.7%**∗

to provoke connection initiation from the victim with the goal of revealing their real MAC address. Considering that such active attack was not the main focus of their work, the authors have concluded from their experiments that in practice one can test only a small number of different SSIDs. Our work however was focused on optimizing the active attack, where we have shown that depending on scanning and idle periods of the Wi-Fi enabled device and the size of the opportunity window, it is possible to test dictionaries more than 10 times bigger in size.

6. Conclusion

In this paper we propose an attacking strategy to extract victim's preferred network list from a mobile device while the device is in Active Service Discovery mode with broadcast scan, or passive scan mode. We introduced the SSID Oracle attack that queries a set of SSIDs from a dictionary against the victim's device by faking APs. We calculated the optimal parameters for the attack execution and proposed a detailed mathematical model depicting the attack.

The model has been confirmed by running extensive tests on different smartphone devices. Furthermore, we also created a simulator to cover more complex attack scenarios with low Wi-Fi channel quality.

Since users are mobile with their devices, the attacker's opportunity window is quite small and the attack depends on having SSIDs which are highly likely to be present in victim's PNL. For that reason we also proposed an example recommender system based algorithm for building a high quality dictionary. We have concluded that by choosing the right dictionary and by executing the attack using our model, it is possible to achieve high probability of success when attempting to disclose an SSID from victim's PNL.

For future work we plan to work on finding solutions that can protect against SSID Oracle attacks. One solution would be to use Geofencing technique in which the device will only respond to probes for APs which are both known and geographically nearby. We plan to further research dynamic tracking and other aspects of Wi-Fi network traffic.

Appendix

A. Mathematical Excerpt

A.1. Hit Chance as a Function of $\boldsymbol{T}$ *and* $\boldsymbol{L}$. The goal of this subsection is to find a mathematical model describing $\mathbb{P}[hit]$ in order to compare it with performed tests, given that the observed chunk holds at least one *SSID* from $\mathscr{P}$. We will be breaking the chunk retrasmission interval $T \in [0, 2T_S + 2T_I]$ to multiple smaller intervals and each of those intervals to multiple smaller intervals τ so that they are easily solvable. Dictated by the fact $T_S \ll T_I$, parts of the equation can then be ignored.

Separate interval solutions follow the general expression for a negated probability that none of n tracked SSID's transmitted during T_S period is caught: $1 - (1 - p)^{\lfloor n \rfloor}(1 - p(n - \lfloor n \rfloor))$. Since one chunk contains one tracked SSID, n also corresponds to the number of fully transmitted chunks during active scanning.

For each T interval the following holds:

$$\begin{aligned}
\mathbb{P}[hit] &= \sum_{\tau_i} \mathbb{P}[hit \mid t \in \tau_i] \cdot \mathbb{P}[t \in \tau_i] \\
\mathbb{P}[t \in \tau_i] &= \frac{|\tau_i|}{T_I + T_S} \\
\mathbb{P}[hit \mid t \in \tau_i] &= \int_{\tau_i} \mathbb{P}[hit \mid t] \cdot p(t)\,dt \\
&= \int_{\tau_i} \mathbb{P}[hit \mid t] \cdot \frac{1}{|\tau_i|} dt \\
\mathbb{P}[hit \mid t] &= 1 - (1-p)^{\lfloor n_i \rfloor} (1 - p(n_i - \lfloor n_i \rfloor))
\end{aligned} \tag{A.1}$$

Note that having $n = 0$ means that during the τ interval no slots will be tested. Also note that having $|\tau_i| \ll (T_S + T_I)$ means that $\mathbb{P}[t \in \tau_i] \approx 0$. We will now move to solving each interval:

(1) $T \le T_S$

$$\begin{aligned}
\tau_1 &\in [0, T_S - T] \quad |\tau_1| \le T_S \\
\tau_2 &\in [T_S - T, T_S] \quad |\tau_2| \le T_S \\
\tau_3 &\in [T_S, T_I + T_S - T] \quad |\tau_3| \approx T_I \\
\tau_4 &\in [T_I + T_S - T, T_I + T_S] \quad |\tau_4| \le T_S \\
n_1 &= \frac{rT}{L} \\
n_2 &= \frac{r(T_S - t)}{L} \\
n_3 &= 0 \\
n_4 &= \frac{r(T + t - T_S - T_I)}{L}
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
\mathbb{P}[hit] &= \sum_{\tau_i} \mathbb{P}[hit \mid t \in \tau_i] \cdot \mathbb{P}[t \in \tau_i] \\
&= (\mathbb{P}[hit \mid t \in \tau_1] + \mathbb{P}[hit \mid t \in \tau_2] + \mathbb{P}[hit \mid t \in \tau_4]) \\
&\quad \cdot \frac{T_S}{T_S + T_I} + \mathbb{P}[hit \mid t \in \tau_3] \cdot \frac{T_I}{T_I + T_S} \\
&\approx 0
\end{aligned} \tag{A.3}$$

($\mathbb{P}[hit \mid t \in \tau_1]$, $\mathbb{P}[hit \mid t \in \tau_2]$, $\frac{T_S}{T_S + T_I}$: ≈ 0; $\mathbb{P}[hit \mid t \in \tau_3]$: 0)

(2) $T_S \le T \le T_I$

$$\begin{aligned}
&\tau_1 \in [0, T_S] \quad |\tau_1| \le T_S \\
&\tau_2 \in [T_S, T_S + T_I - T] \\
&\tau_3 \in [T_S + T_I - T, 2T_S + T_I - T] \quad |\tau_3| \le T_S \\
&\tau_4 \in [2T_S + T_I - T, T_S + T_I] \quad |\tau_4| = T - T_S \\
&n_4 = \frac{rT_S}{L}
\end{aligned} \tag{A.4}$$

$$\begin{aligned}
\mathbb{P}[hit] &\approx \mathbb{P}[hit \mid t \in \tau_4] \cdot \frac{T - T_S}{T_S + T_I} \\
&\overset{(1)}{=} \left(1 - (1-p)^{\lfloor n_4 \rfloor} (1 - p(n_4 - \lfloor n_4 \rfloor))\right) \cdot \frac{T - T_S}{T_S + T_I}
\end{aligned} \tag{A.5}$$

(3) $T_I \le T \le T_S + T_I$

$$\begin{aligned}
&\tau_1 \in [0, T_S + T_I - T] \quad |\tau_1| \le T_S \\
&\tau_2 \in [T_S + T_I - T, T_S] \quad |\tau_2| \le T_S \\
&\tau_3 \in [T_S, 2T_S + T_I - T] \quad |\tau_3| \le T_S \\
&\tau_4 \in [2T_S + T_I - T, T_S + T_I] \quad |\tau_4| = T_I - T_S \\
&n_4 = \frac{rT_S}{L}
\end{aligned} \tag{A.6}$$

$$\begin{aligned}
\mathbb{P}[hit] &\approx \mathbb{P}[hit \mid t \in \tau_4] \cdot \frac{T_I - T_S}{T_S + T_I} \\
&\overset{(1)}{=} \left(1 - (1-p)^{\lfloor n_4 \rfloor} (1 - p(n_4 - \lfloor n_4 \rfloor))\right) \\
&\quad \cdot \frac{T_I - T_S}{T_S + T_I}
\end{aligned} \tag{A.7}$$

(4) $T_S + T_I \le T \le 2T_S + T_I$

$$\begin{aligned}
&\tau_1 \in [0, 2T_S + T_I - T] \quad |\tau_1| \le T_S \\
&\tau_2 \in [2T_S + T_I - T, T_S] \quad |\tau_2| \le T_S \\
&\tau_3 \in [T_S, 2T_S + 2T_I - T] \quad |\tau_3| \approx T_I \\
&\tau_4 \in [2T_S + 2T_I - T, T_S + T_I] \quad |\tau_4| \le T_S \\
&n_3 = \frac{rT_S}{L}
\end{aligned} \tag{A.8}$$

$$\begin{aligned}
\mathbb{P}[hit] &\approx \mathbb{P}[hit \mid t \in \tau_4] \cdot \frac{T_S + 2T_I - T}{T_S + T_I} \\
&\overset{(1)}{=} \left(1 - (1-p)^{\lfloor n_3 \rfloor} (1 - p(n_3 - \lfloor n_3 \rfloor))\right) \\
&\quad \cdot \frac{T_S + 2T_I - T}{T_S + T_I}
\end{aligned} \tag{A.9}$$

(5) $2T_S + T_I \le T \le T_S + 2T_I$

$$\begin{aligned}
&\tau_1 \in [0, T_S] \quad |\tau_1| \le T_S \\
&\tau_2 \in [T_S, T_S + 2T_I - T] \quad |\tau_2| \in [0, T_I - 2T_S] \\
&\tau_3 \in [T_S + 2T_I - T, 3T_S + 2T_I - T] \\
&\qquad\qquad |\tau_3| \approx 2T_S \\
&\tau_4 \in [3T_S + 2T_I - T, T_S + T_I] \quad |\tau_4| \in [0, T_I] \\
&n_2 = \frac{rT_S}{L} \\
&n_4 = 2\frac{rT_S}{L}
\end{aligned} \tag{A.10}$$

$$\begin{aligned}\mathbb{P}[hit] &\approx \mathbb{P}[hit \mid t \in \tau_4] \cdot \frac{T - T_I - 2T_S}{T_S + T_I} \\ &\overset{(1)}{=} \left(1 - (1-p)^{\lfloor n_2 \rfloor}(1 - p(n_2 - \lfloor n_2 \rfloor))\right) \\ &\quad \cdot \frac{2T_I - T}{T_S + T_I} \\ &\quad + \left(1 - (1-p)^{\lfloor n_4 \rfloor}(1 - p(n_4 - \lfloor n_4 \rfloor))\right) \\ &\quad \cdot \frac{T - T_I - 2T_S}{T_S + T_I}\end{aligned} \tag{A.11}$$

Data Availability

In the manuscript, we have described the algorithms as well as the methods we used for collecting and testing the dataset. However, the collected data would violate the privacy of end users and should not be made publicly available.

Disclosure

Bojan Vujatović is now at Google DeepMind, London, UK.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] "Hybrid positioning," accessed: 2017-02-28 https://en.wikipedia.org/wiki/Hybrid_positioning_system.

[2] "Cell tower trilateration," accessed: 2017-02-19 https://wrongfulconvictionsblog.org/2012/06/01/cell-tower-triangulation-how-it-works/.

[3] N. Marques, F. Meneses, and A. Moreira, "Combining similarity functions and majority rules for multi-building, multi-floor, WiFi positioning," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN '12)*, pp. 1–9, Sydney, Australia, November 2012.

[4] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, and J. Fangmeyer Jr., "Evolution of Indoor Positioning Technologies: A Survey," *Journal of Sensors*, vol. 2017, Article ID 2630413, 21 pages, 2017.

[5] J. Duque Domingo, C. Cerrada, E. Valero, and J. A. Cerrada, "Indoor Positioning System Using Depth Maps and Wireless Networks," *Journal of Sensors*, vol. 2016, Article ID 2107872, 8 pages, 2016.

[6] L. Sun, S. Chen, Z. Zheng, and L. Xu, "Mobile Device Passive Localization Based on IEEE 802.11 Probe Request Frames," *Mobile Information Systems*, vol. 2017, Article ID 7821585, 10 pages, 2017.

[7] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, and J. Stefa, "Signals from the crowd: Uncovering social relationships through smartphone probes," in *Proceedings of the 13th ACM Internet Measurement Conference, IMC 2013*, pp. 265–276, Spain, October 2013.

[8] A. Di Luzio, A. Mei, and J. Stefa, "Mind your probes: De-anonymization of large crowds through smartphone WiFi probe requests," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016*, San Francisco, CA, USA, April 2016.

[9] M. Cunche, M.-A. Kaafar, and R. Boreli, "Linking wireless devices using information contained in Wi-Fi probe requests," *Pervasive and Mobile Computing*, vol. 11, pp. 56–69, 2014.

[10] M. Cunche, . Mohamed Ali Kaafar, and R. Boreli, "I know who you will meet this evening! Linking wireless devices using Wi-Fi probe requests," in *Proceedings of the 2012 IEEE Thirteenth International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pp. 1–9, San Francisco, CA, USA, June 2012.

[11] B. Bonné, A. Barzan, P. Quax, and W. Lamotte, "WiFiPi: involuntary tracking of visitors at mass events," in *Proceedings of the IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '13)*, pp. 1–6, Madrid, Spain, June 2013.

[12] M. Chernyshev, C. Valli, and P. Hannay, "On 802.11 access point locatability and named entity recognition in service set identifiers," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 3, pp. 584–593, 2016.

[13] F. Zhao, W. Shi, Y. Gan, Z. Peng, and X. Luo, "A localization and tracking scheme for target gangs based on big data of Wi-Fi locations," *Cluster Computing*, pp. 1–12, 2018.

[14] "Wireless Geographic Logging Engine," accessed: 2017-02-28, https://wigle.net/.

[15] "What are Active and Passive scanning," accessed: 2017-02-28, http://www.wi-fi.org/knowledge-center/faq/what-are-passive-and-active-scanning.

[16] J. Freudiger, "Short: How talkative is your mobile device? An experimental study of Wi-Fi probe requests," in *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2015*, June 2015.

[17] N. Sidiropoulos, M. Mioduszewski, P. Oljasz, and E. Schaap, "Open Wifi SSID Broadcast vulnerability," *SSN Project Assessment*, 2012.

[18] "Karma tool," accessed: 2017-02-19, https://www.offensive-security.com/kali-linux/kali-linux-evil-wireless-access-point/.

[19] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC address randomization is not enough: an analysis of Wi-Fi network discovery mechanisms," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 413–424, ACM, Xi'an, China, June 2016.

[20] V. Ramachandran, "Scapy: Dictionary Attacks on Hidden SSID Networks," accessed: 2017-02-19, http://www.pentesteracademy.com/video?id=471.

[21] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Transactions on Information and System Security*, vol. 22, no. 1, pp. 143–177, 2004.

[22] S. Jamil, A. Basalamah, S. Khan, and A. Lbath, "Classifying smartphone screen ON/OFF State Based on WiFi probe patterns," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2016*, pp. 301–304, Germany, September 2016.

[23] W. Wang, R. Joshi, A. Kulkarni, W. K. Leong, and B. Leong, "Feasibility study of mobile phone WiFi detection in aerial search and rescue operations," in *Proceedings of the 4th Asia-Pacific Workshop on Systems, APSys 2013*, Singapore, July 2013.

[24] X. Hu, L. Song, D. Van Bruggen, and A. Striegel, "Is there WiFi yet? How aggressive probe requests deteriorate energy and throughput," in *Proceedings of the ACM Internet Measurement Conference, IMC 2015*, pp. 317–323, Japan, October 2015.

[25] Pieter Robyns, Bram Bonné, Peter Quax, and Wim Lamotte, "Noncooperative 802.11 MAC Layer Fingerprinting and Tracking of Mobile Devices," *Security and Communication Networks*, vol. 2017, Article ID 6235484, 21 pages, 2017.

[26] M. Cunche, "I know your MAC address: targeted tracking of individual using Wi-Fi," *Journal of Computer Virology and Hacking Techniques*, vol. 10, no. 4, pp. 219–227, 2014.

[27] G. Shi and K. Li, *Signal Interference in WiFi and ZigBee networks*, 2017, accessed: 2017-02-19, https://books.google.hr/books?id=rAFSDQAAQBAJ.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pp. 285–295, ACM, New York, NY, USA, 2001.

[29] Aircrack, "Aircrackng," accessed: 2017-02-19, https://www.aircrack-ng.org/.

[30] Y. S. Kim, Y. Tian, L. T. Nguyen, and P. Tague, "LAPWiN: Location-aided probing for protecting user privacy in Wi-Fi networks," in *Proceedings of the 2014 IEEE Conference on Communications and Network Security, CNS 2014*, pp. 427–435, USA, October 2014.

[31] M. V. Barbera, A. Epasto, A. Mei, S. Kosta, V. C. Perta, and J. Stefa, "CRAWDAD dataset sapienza/probe-requests (v. 2013-09-10)," Sep. 2013, Downloaded from https://crawdad.org/sapienza/probe-requests/20130910.

[32] Abu-Mostafa, S. Yaser, Magdon-Ismail, Malik, Lin, and Hsuan-Tien, "Learning From Data," in *AMLBook*, 2012.

[33] N. Sidiropoulos, M. Mioduszewski, P. Oljasz, and E. Schaap, *Open Wifi SSID Broadcast vulnerability*, 2012.

[34] K. Sethom and H. Afifi, "Requirements and adaptation solutions for transparent handover between wifi and Bluetooth," in *Proceedings of the 2004 IEEE International Conference on Communications*, pp. 3916–3920, France, June 2004.

[35] A. E. Redondi and M. Cesana, "Building up knowledge through passive WiFi probes," *Computer Communications*, vol. 117, pp. 1–12, 2018.

[36] S. Seneviratne, F. Jiang, M. Cunche, and A. Seneviratne, "SSIDs in the wild: Extracting semantic information from WiFi SSIDs," in *Proceedings of the 2015 IEEE 40th Conference on Local Computer Networks, LCN 2015*, pp. 494–497, USA, October 2015.

[37] C. Matte and M. Cunche, "Beam me up, Scotty: identifying the individual behind a MAC address using Wi-Fi geolocation spoofing," in *Proceedings of the 1er Colloque sur la Confiance Numérique en Auvergne*, 2014.

[38] D. Goodin, "No, this isn't a scene from Minority Report. This trash can is stalking you," accessed: 2017-02-28, http://arstechnica.com/security/2013/08/no-this-isnt-a-scene-from-minority-report-this-trash-can-is-stalking-you/.

[39] "DIY stalker boxes spy on Wi-Fi users cheaply and with maximum creep value," accessed: 2017-02-28, http://arstechnica.com/security/2013/08/diy-stalker-boxes-spy-on-wi-fi-users-cheaply-and-with-maximum-creep-value/.

[40] T. Kropeit, *Don't Trust Open Hotspots: Wi-Fi Hacker Detection and Privacy Protection via Smartphone [Bachelor's Thesis]*, Ruhr-Universität-Bochum, 2015.

[41] C. Matte, M. Cunche, F. Rousseau, and M. Vanhoef, "Defeating MAC address randomization through timing attacks," in *Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2016*, pp. 15–20, July 2016.