Capturing Privacy-preserving User Contexts with IndoorHash

Lakhdar Meftah, Romain Rouvoy, Isabelle Chrisment

Capturing Privacy-preserving User Contexts with IndoorHash

Lakhdar Meftah[1], Romain Rouvoy[2], and Isabelle Chrisment[3]

[1] Inria / Univ. Lille, France
lakhdar.meftah@inria.fr
[2] Univ. Lille / Inria / IUF, France
romain.rouvoy@univ-lille.fr
[3] LORIA-TELECOM Nancy / Univ. Lorraine, France
isabelle.chrisment@loria.fr

Abstract. IoT devices are ubiquitous and widely adopted by end-users to gather personal and environmental data that often need to be put into context in order to gain insights. In particular, location is often a critical context information that is required by third parties in order to analyse such data at scale. However, sharing this information is *i)* sensitive for the user privacy and *ii)* hard to capture when considering indoor environments.
This paper therefore addresses the challenge of producing a new location hash, named IndoorHash, that captures the indoor location of a user, without disclosing the physical coordinates, thus preserving their privacy. This location hash leverages surrounding infrastructure, such as WiFi access points, to compute a key that uniquely identifies an indoor location.
Location hashes are only known from users physically visiting these locations, thus enabling a new generation of privacy-preserving crowdsourcing mobile applications that protect from third parties re-identification attacks. We validate our results with a crowdsourcing campaign of 31 mobile devices during one month of data collection.

Keywords: Location Hash · Mobile Computing · User Privacy.

1 Introduction

In order to meet the user expectation, mobile apps are supposed to understand the user environment and act accordingly. To better capture the surrounding context, these mobile apps rely on data gathered from embedded sensors and user location data sits at the top of the list of key context information. However, the processing of raw location data may leak some privacy-sensitive knowledge that the end user might not accept. Furthermore, embedded location sensors (*e.g.*, GPS) fail at locating the end user once indoor, which seriously prevents the deployment of location-based services at a small scale (*e.g.*, offering location-based services in a mall).

To address these two challenges, we propose a new data structure to accurately capture the indoor location of end users without disclosing their physical location. By inferring such a logical location or place, we intend to leverage the development of location-based services that can work in indoor environments without exposing the user privacy. More specifically, our contribution consists in the definition of a new similarity hash function (also known as *simhash*), named IndoorHash, which encodes the indoor location of an end user in a privacy-preserving way. Interestingly, this IndoorHash is robust to re-identification attacks, as the physical location cannot be inferred by an adversary, but allows location-based services to compare the similarity of locations through pairwise comparisons. In this paper, we report on the robustness and accuracy of IndoorHash built from WiFi scan data, which is a lightweight contextual information to collect. As a matter of evaluation, we embedded IndoorHash in a mobile app that has been deployed during one month to capture the daily routine of 31 users. Our results show that IndoorHash succeeds to accurately capture the location of end users by inferring stable logical places, while preserving their privacy.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 introduces our implementation of a privacy-preserving indoor location hash, named IndoorHash. Section 4 evaluates IndoorHash along an empirical deployment we conducted. Section 5 illustrates the benefits of IndoorHash for building privacy-sensitive location-based services. Finally, Section 6 concludes on this work.

2 Related Work

Mobile devices are equipped with a wide panel of embedded sensors that mobile apps can use to acquire key insights about the surrounding environment and the user activities. Among the context information of critical importance, GPS location and WiFi networks are common data that are heavily exploited to locate the user. More specifically, when it comes to indoor location, most of *indoor positioning systems* and *WiFi fingerprinting systems* use the surrounding scanned WiFi access points to locate users indoor or to track users' places.

2.1 WiFi Indoor Positioning Systems

Due to the absence of the *Global Positioning System* (GPS) signal inside buildings, many systems have been proposed as an indoor positioning system [12]. Among these systems, WiFi-based systems, which take advantage of the legacy WiFi *Access Points* (APs) to estimate the location of a mobile device down to 1.5 meters precision using the WiFi signal strength [14]. While some research works require a prior training phase (online mode) to map the location of the antennas [26, 36], other proposals have been proposing an automatic way to build a radio map without any prior knowledge about the antennas [32, 18, 16]. In particular, Liu *et al.* [17] propose a peer-assisted localization method to improve

localization accuracy. Li *et al.* [15] introduce a privacy-preserving WiFi indoor localization system. However, they argue that the localization query can inevitably leak the client location and lead to potential privacy violations. Jin *et al.* [13] propose a real-time WiFi positioning algorithm with the assistance of inertial measurement unit to overcome the *Received Signal Strength* (RSS) variation problem. Pulkkinen *et al.* [26] present an automatic fingerprinting solution using theoretical properties of radio signals, they rely on the locations of WiFi APs and collecting training measurements. Salamah *et al.* [29] use the *Principle Component Analysis* (PCA) to reduce the computation cost of the WiFi indoor localization systems based on machine learning approach. Yiu *et al.* [37] apply training measurements and a combined likelihood function from multiple APs to measure the indoor and the outdoor user position. Capurso *et al.* [4] present an indoor and outdoor detection mechanism, which can be used to optimize GPS energy usage. Yiu *et al.* [36] review the various methods to create the radiomap. Then, they examined the different aspects of localization performance like the density of WiFi APs and the impact of an outdated radiomap. Ahmed *et al.* [1] provide a new optimized algorithm for fast indoor localization using WiFi channel state information. Caso *et al.* [5] introduce an indoor positioning system that relies on the RSS to generate a discrete RSS radiomap. Li *et al.* [16] present SoiCP, a seamless outdoor-indoor crowdsourcing positioning system without requiring site surveying. Crowdsourced WiFi signals are used to build a radiomap without any prior knowledge.

Synthesis. Indoor location systems leveraging WiFi scans are proven to be very useful and accurate. However, existing indoor location systems remain limited to a restricted number of buildings, which have to be either equipped specifically to locate users from WiFi APs, or have to consider a large number of users to reduce the location errors. Most approaches propose to learn the position of these WiFi antennas for every building. In our work, we do not require any prior knowledge about the building and the location of the WiFi antennas. Moreover, we do not intend to physically locate the user, but to capture her context by observing if she is located in a different place or not, which can be answered without an *a priori* knowledge of a map. Then, we consider the related work that uses WiFi fingerprinting to capture the user context.

2.2 WiFi Fingerprinting Systems

Given the attractive cost of considering WiFi signals instead of GPS [38], the state of the art has been extensively considering WiFi signals to capture both indoor and outdoor user contexts. Surrounding WiFi APs can be exposed to mobile apps and act as a place fingerprint—different WiFi AP signals received thus result in different places. As such, using WiFi scans, one can track end users for several hours and obtain valuable information about activities [24], personality traits [23] and routines [27]. However, these WiFi scans can also be used to establish a unique user fingerprint [8, 39], extract social networks [33] or track

users offline [31, 35]. Furthermore, by knowing the location of WiFi APs, the location of users can be inferred from their fingerprint. In particular, some research works are focusing on optimizing the similarity distance between two WiFi fingerprints [2, 9, 20, 37]. Zhang *et al.* [40] introduce POLARIS, a location system using cluster-based solution for WiFi fingerprints. They explain how to collect and compress city-scale fingerprints, and then, how to find the location of a user using similar WiFi fingerprints. Sakib *et al.* [28] present a method to contextualize the mobile user using the WiFi APs fingerprints, with a clustering algorithm that creates a place for each group of WiFi APs. They validate their results with cellular cell ids datasets. Sapiezynski *et al.* [30] use WiFi to enhance GPS traces composed of one GPS location per day, they can locate 80% of mobility across the population. Wind *et al.* [34] use WiFi scans to infer stop locations for users, their algorithm can tell if a user is moving or in a stationary state using just WiFi fingerprints. Finally, Choi *et al.* [7] propose a method for an energy efficient WiFi scanning for user contextualization, they try to minimize the number of scans depending on the scanned WiFi APs.

Synthesis. While a lot of work have been done to localize the user using WiFi indoor location techniques, most of them require either a training phase or user input to locate the user using the WiFi antennas. Several research works have been focusing on the WiFi fingerprinting based on the clustering algorithms and similarity distances, but they do not report on the utility of the collected data, its cost or its privacy-preserving methods. Furthermore, they do not provide any library or framework in order for their methods to be adopted, evaluated and further improved.

3 Introducing IndoorHash

This section introduces our current design of INDOORHASH and the properties we build on to control the number of relevant hashes.

3.1 Computing an IndoorHash

To design our INDOORHASH, we implement a modified version of the SimHash algorithm [6]. SimHash is a technique for quickly estimating how similar two sets are. This algorithm was reported to be used by the Google Crawler to find near duplicate pages.

In the context of INDOORHASH, we tokenize the words from the SimHash algorithm as pairs of digits in the hexadecimal MAC address format—*e.g.*, we convert the MAC address `00:FF:AB:0F:C2:C7` into the set of words $\{\langle \texttt{00} : 1\rangle, \langle \texttt{FF} : 1\rangle, \langle \texttt{AB} : 1\rangle, \langle \texttt{0F} : 1\rangle, \langle \texttt{C2} : 1\rangle, \langle \texttt{C7} : 1\rangle\}$. Then, the input text we use is the list of all WiFi APs MAC addresses that is returned when triggering a WiFi scan. This list is therefore converted into a set of unique words weighted by the apparition frequencies resulting. For example, a list of 3 MAC addresses can result in the following input set $\{\langle \texttt{00} : 3\rangle, \langle \texttt{FF} : 2\rangle, \langle \texttt{AB} : 2\rangle, \langle \texttt{0F} : 3\rangle, \langle \texttt{C2} : 2\rangle, \langle \texttt{C7} :$

$1\rangle, \langle \texttt{C7} : 1\rangle, \langle \texttt{D0} : 1\rangle, \langle \texttt{02} : 1\rangle, \langle \texttt{DD} : 1\rangle, \langle \texttt{BE} : 1\rangle\}$. From this input set, we compute a SimHash whose *size* can be defined according to the targeted precision (cf. Algorithm 1).

Algorithm 1 IndoorHash Algorithm.

```
function TokenizeScan(scan)
    tokens ← ∅
    for all ⟨bssid⟩ ∈ scan do
        words ← Split(bssid, ' : ')
        for all word ∈ words do
            if word ∈ tokens then
                frequency ← GetFrequency(tokens, word)
                tokens ← tokens ∪ ⟨word : frequency + 1⟩
            else
                tokens ← tokens ∪ ⟨word : 1⟩
            end if
        end for
    end for
    return tokens
end function

function ComputeIndoorHash(scan, size)
    tokens ← TokenizeScan(scan)
    return SimHash(tokens, size)
end function
```

3.2 Comparing an IndoorHash

Unlike standard hashing algorithms, hashes computed by the SimHash algorithm can be compared to estimate the similarity of their respective input sets. To compute this similarity, IndoorHash computes the Hamming distance between the two input hashes to report on the ratio of similar bits (cf. Algorithm 2).

Algorithm 2 IndoorHash Similarity.

```
function CompareIndoorHash(hash1, hash2)
    distance ← HammingDistance(hash1, hash2)
    sim ← 1 − (distance/Size(hash1))
    if sim < 0.5 then
        return 0
    end if
    return (sim − 0.5) × 2                          ▷ Expand the similarity score
end function
```

However, applying standard SimHash similarity on IndoorHash tends to report a ratio above 0.50, due to the high probability of sharing words between dissimilar MAC addresses. We therefore chose to expand our similarity range, and only consider the values between 0.50 and 1.0 as relevant similarity values. This approach improves the sensibility of the IndoorHash when comparing their values.

3.3 Indexing an IndoorHash

We store an IndoorHash in a *K-Nearest Neighbors* (KNN) graph that organizes all the computed IndoorHash according to their pairwise similarity (cf. Algorithm 3). The IndoorHash storage procedure searches and binds to the closest neighbors in the KNN graph. If the most similar IndoorHash is already connected to k other hashes, the farest hash is propagated among the remaining neighbors.

Algorithm 3 IndoorHash Storage.

```
procedure StoreIndoorHash(from,hash)
    similarity ← CompareIndoorHash(hash, from)
    neighbors ← GetKnnNeighbors(from)
    maxsim, simnode ← Max(hash, neighbors, CompareIndoorHash)
    if similarity < maxsim then ▷ Propagates hash to the most similar neighbor
        StoreIndoorHash(hash, simnode)
    else                                          ▷ from is the most similar hash
        k ← Size(neighbors)
        if k < K_MAX then
            AddKnnNeighbour(from, hash)
        else                                ▷ Replaces the least similar neighbor by hash
            minsim, node ← Min(hash, neighbors, CompareIndoorHash)
            RemoveKnnNeighbour(from, node)
            AddKnnNeighbour(from, hash)
            StoreIndoorHash(from, node)
        end if
    end if
end procedure
```

This indexing structure allows to quickly search for a similar IndoorHash in the KNN graph by starting from any random node in this graph and converging through the closest neighbors—*i.e.*, most similar IndoorHash in our case. The similarity search (SimSearch) returns a node if the similarity of the closest neighbor with the input IndoorHash is above an expected similarity *threshold* (cf. Algorithm 4).

Algorithm 4 IndoorHash Search.

```
function SimSearch(storage, hash, threshold)
    seed ← RandomNode(storage)
    simnode ← SearchIndoorHash(seed, hash, ratio)
    similarity ← CompareIndoorHash(hash, simnode)
    if similarity >= threshold then
        return simnode
    end if
    return NONE
end function

function SearchIndoorHash(from, hash)
    similarity ← CompareIndoorHash(from, hash)
    neighbors ← GetKnnNeighbors(from)
    maxsim, simnode ← Max(hash, neighbors, CompareIndoorHash)
    if similarity > maxsim then              ▷ from is the most similar neighbor
        return from
    end if
    return SearchIndoorHash(simnode, hash)
end function
```

3.4 Classifying an IndoorHash

IndoorHash leverages the SimHash algorithm [6] to generate a robust hash from a list of visible WiFi APs. Depending on situations, we can consider that:

1. IndoorHash refers to a *new hash*—*i.e.*, whenever a hash is stable for more than $5mn$, a new IndoorHash is stored locally,
2. IndoorHash refers to a *known hash* when it is stored locally,
3. IndoorHash refers to a *commuting hash* when no hash can be computed or the hash keeps changing after every scan.

Algorithm 5 describes how we classify an IndoorHash along the above situations. This context classification algorithm allows not only the device to adjust its behavior accordingly, but also avoid an explosion of the number of collected IndoorHash. Indeed, by setting an appropriate `DELAY` and `SIMILARITY` threshold, one can filter out any candidate IndoorHash that is considered as irrelevant (*e.g.*, spending less than $5mn$ in a place does not require to be remembered). By adjusting the reference *storage*, the algorithm can leverage a graph of shared location hashes to quickly classify it as a known location, without waiting for the classification delay to be elapsed.

3.5 Sharing an IndoorHash

IndoorHash does not make any strong assumption on the mobile device that is used to capture the current indoor location. As such, it can therefore be shared among users to ease the detection of known locations, and even used as a location

Algorithm 5 IndoorHash Classification.

```
function ClassifyIndoorHash(storage, timestamp, hash)
    if hash <> lastHash then
        lastHash ← hash
        lastTimestamp ← timestamp
        return COMMUTING_HASH
    else
        found ← SimSearch(storage, hash, SIMILARITY)          ▷ Similarity search
        if found then
            return KNOWN_HASH
        else
            if timestamp − lastTimestamp > DELAY then
                StoreIndoorHash(storage, hash)
                return KNOWN_HASH
            end if
            return NEW_HASH
        end if
    end if
end function
```

key to store some context information. For example, whenever an IndoorHash is computed, a device can query a cloud service to obtain some information associated to this location. The remote service can adjust the similarity score to constrain how similar two indoor locations should be in order to share the associated information, thus avoiding adversaries accessing any information shared online. Conversely, the remote storage service only manipulates IndoorHash and cannot learn about the physical location of the end user, thus avoiding any leak of sensitive geolocated information (cf. Section 4.1). Furthermore, at scale, the remote storage of known IndoorHash can leverage the indexing structure introduced in Section 3.3 to quickly find the most similar IndoorHash among the graph of known hashes shared online.

4 Empirical Evaluation

This section reports on the empirical evaluation we conducted to assess IndoorHash as a robust indoor location hash that can be used to capture the current indoor location of a user, while preserving her privacy. We therefore start by conducting a privacy analysis of our contribution, before introducing the DayKeeper app we deployed to assess the accuracy of IndoorHash to capture indoor locations.

4.1 Privacy Analysis

Recover the GPS location of IndoorHash. Existing WiFi fingerprinting methods exploit raw MAC addresses to recover a GPS location from crowdsourced

datasets that share key-value pairs of MAC addresses and the associated GPS location [30]. In our case, IndoorHash shares the hash of a set of MAC addresses. Hashing a single MAC address is not enough to preserve the user privacy, as a rainbow table could be created for all the hashes. In our solution, we therefore hash all detected MAC addresses of WiFi APs, which adds a huge entropy to the generated hash and therefore reduces the risk to infer the physical location captured by a given IndoorHash. Concretely, to recover all the possible hashes, a rainbow table would have to be created for 240 bits—when assuming hashes built from 5 MAC addresses (48 bits)—which is far beyond the state of the art. Additionally, as IndoorHash does not make any assumption on the number of input MAC addresses, such an attack becomes impracticable.

Identify users from clusters of IndoorHash. As no GPS location can be recovered from an IndoorHash, the state-of-the-art geo-spatial attacks, like the ST-DBSCAN [3], fail to cluster the user locations. Therefore, running a similar attack on a set of IndoorHash can only produce clusters of hashes grouped by similarity, which fails to disclose any sensitive information about the user. Indeed, even though an IndoorHash can be considered as *Point of Interest* (POI), no metata (label, location, category, etc.) can be exploited by an adversary to infer some privacy sensitive knowledge.

Conclusion. IndoorHash provides a robust location hash that captures the physical context of a user, without disclosing any privacy sensitive information regarding the locations she visited. Yet, the adoption of IndoorHash does not prevent developers from carefully anonymizing any payload that could be attached to such a hash (*e.g.*, IP address, device model, user identifier) to avoid indirect privacy breach.

4.2 DayKeeper Android App

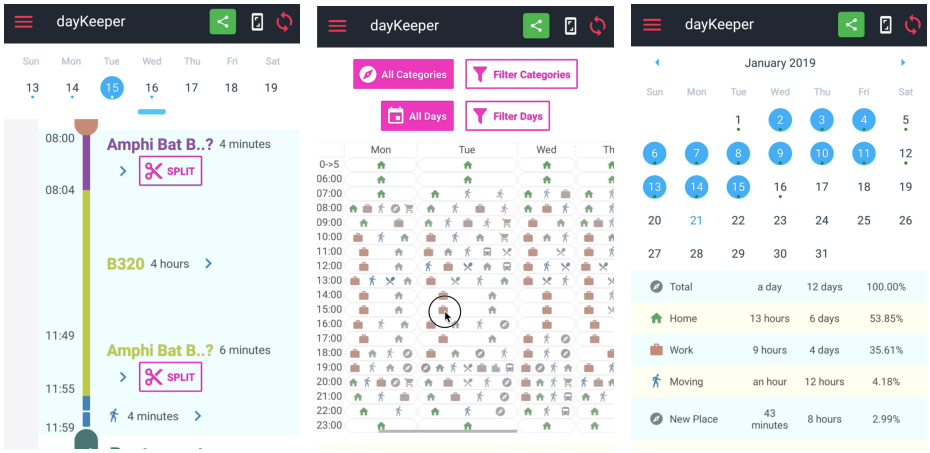
To demonstrate and assess IndoorHash in the wild, we developed an Android app that embeds our IndoorHash as a software library. This Android app, named DayKeeper, aims at keeping track of the daily activities of end users in order to report on the time they spent in different locations (home, office, shopping, leisure). When users opt in, IndoorHash and some metrics (cf. Section 4.3) are also periodically synchronized on a remote server, using the APISENSE platform [10], for *postmortem* analysis purpose. This data collection campaign conforms to the *Institutional Review Board* (IRB) approval associated to the exploitation of the APISENSE platform.[4]

Figure 1 depicts some screenshots of the DayKeeper Android app.[5] In particular, users of DayKeeper are free to navigate through daily (cf. Figure 1a), weekly (cf. Figure 1b) and monthly views (cf. Figure 1c) and to eventually label the reported IndoorHash to keep track of their personal timeline. With

[4] https://apisense.io

[5] Available from https://play.google.com/store/apps/details?id=io.apisense.netquality

DayKeeper, we aim at demonstrating that IndoorHash can offer a fine grain analysis of user activities by splitting different activities observed in a single building (*e.g.*, office, meeting, cafetaria, gym). Furthermore, as DayKeeper leverages the WiFi APs, it provides a lightweight activity logger that does not drain the device battery by continuously requesting the GPS sensor.



(a) Daily view (b) Weekly view (c) Monthly view

Fig. 1: Screenshots of the DayKeeper Android App.

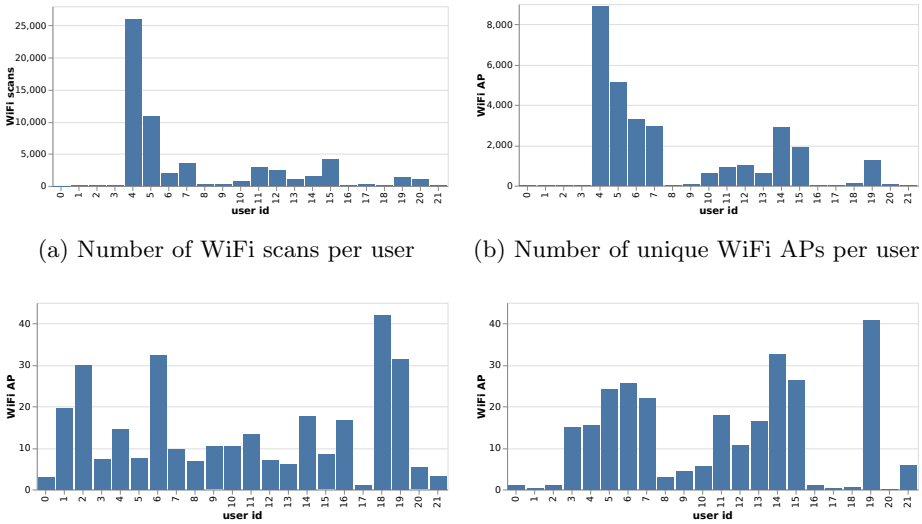
The IndoorHash is continuously updated by DayKeeper using a background service that periodically scans the surrounding WiFi APs, computes the associated IndoorHash (cf. Section 3.1) and classifies it with regards to *known hashes* (cf. Section 3.4). In DayKeeper, *known hashes* are reflected as a continuous timeline of a given color, while *commuting hashes* are displayed as a transition 'Moving' between two *known hashes*.

4.3 Deployment Statistics

Overall, we can report that the DayKeeper has been installed by 31 users along one month. During this period, users have detected 58,500 unique WiFi APs, resulting in 12,201 IndoorHash. The remainder of this section covers the statistics we collected along our experimentation period (cf. Figure 2). One should note that we filtered out 9 users of DayKeeper who used our mobile app, but did not opt in the sharing of their metrics. As users come and go and can activate or deactivate the context acquisition, the data contributed by each user varies greatly. Figure 2a therefore depicts the number of WiFi scans triggered per user as a more representative indicator of the activity of registered users.

Then, Figure 2b reports on the number of unique WiFi AP per user, this number reflects the diversity of locations that have been visited by a user: the

more the user moves, the more WiFi APs. Figure 2c shares some indications on the number of WiFi APs that can be captured along a scan. This number depends on the density of WiFi APs exposed to a user, but it clearly shows that users are often exposed to more than 5 MAC addresses, thus strengthening the resilience of IndoorHash against rainbow table attacks (cf. Section 4.1). Finally, Figure 2d reports the average number of WiFi APs per hour for each user, thus giving a clear signal about the continuous WiFi coverage of users.



(a) Number of WiFi scans per user

(b) Number of unique WiFi APs per user

(c) Average number of WiFi APs per scan per user

(d) Average number of WiFi AP per hour

Fig. 2: DayKeeper App Deployment Statistics.

Overall, Figures 2 delivers some insightful feedback about the diversity of profiles of DayKeeper users—*e.g.*, whether they are moving or stationary. For example, we can observe that the user with the identifier 20 was mostly stationary, as her device contributed a lot of hours (cf. Figure 2a), but with only 57 unique WiFi APs (cf. Figure 2b) and an average of 0.10 WiFi APs per hour (cf. Figure 2d). On the other side, the user with identifier 19 has been using the DayKeeper app for a shorter period of time, but exhibiting a more active profile. The rest of our evaluation considers all the users who have been sharing their IndoorHash with our remote storage server, no matter their profile.

4.4 IndoorHash Evaluation

This section focuses on a quantitative evaluation of the IndoorHash. To assess our contribution, we compare it to alternative hashing and similarity algorithms adopted by the state of the art, as described below.

Evaluation metrics. In the context of this evaluation, we focus on assessing the robustness of the location hash to accurately capture the context of a user. We therefore consider the following metrics:

- **Total period** represents the total number of hours of activities recognized by the algorithms for all of the users (the higher the better). This metrics reflects the capability of the algorithm to classify the context of the user;
- **Max period** reflects the maximum number of hours assigned to a single location hash (the higher the better). This metrics highlights the capability of each algorithm to detect known locations;
- **Max occurrences** represents the maximum number of occurrences that can be observed for one location hash (the higher the better). This metrics demonstrate the capability of the algorithms to infer similar location hashes for recurrent locations;
- **Hashes with 1+ hour** represents the number of location hashes that last for more than one hour (the higher the better). This metrics highlights the capability of the location hash to capture stationary conditions;
- **Max detected hashes** represents the maximum number of distinct location hashes for a given location (the lower the better). This metrics reflects the quality of the generated location hashes.

Hashing and similarity algorithms. We start by comparing IndoorHash with the most-related similarity algorithms:

- The Jaccard index is the baseline algorithm that we apply to compare raw scans including plain MAC addresses of detected WiFi APs,
- The SimHash correspond to the standard SimHash algorithm [6] applied to plain MAC addresses,
- The IndoorHash refers to the approach described in this paper.

For all the algorithms, we consider a similarity threshold of $0.60f$ as a conservative policy to classify the inferred location hashes.

Table 1: Comparison of alternative hashing and similarity metrics

Variant	Overall period	Max period	Max occurrences	Hashes of 1+ hour	Max detected hashes
Jaccard	2,077	524	3,684	58	22
SimHash	1,112	350	3,127	50	63
IndoorHash	**2,022**	**524**	**3,684**	**59**	**23**

Overall, we can observe in Table 2 that IndoorHash competes with the Jaccard index while providing stronger privacy guarantees. The missing 55 hours that are not recognized by IndoorHash refers to short periods of time ($< 5minutes$) where the recognition algorithm of IndoorHash considers the location as a new hash (cf. Algorithm 5).

Impact of MAC addresses. To further evaluate IndoorHash, we consider 3 variants of IndoorHash to evaluate the impact of the number of MAC addresses

on the stability of the computed hashes. Table 2 shows the metrics for the following variants:

- **No limit** refers to the baseline IndoorHash that does not limit the number of MAC addresses to be included in the computation of the hash;
- **10 MAC** is a variant that limits the computation of the IndoorHash to the first 10 MAC addresses return by the scan;
- **5 MAC** is a variant that limits the IndoorHash computation to the first 5 MAC addresses.
- **1 MAC** is the extreme variant that only consider the first MAC address to compute an IndoorHash.

Table 2: Impact of the MAC addresses on the stability of IndoorHash.

Variant	Overall period	Max period	Max occurrences	Hashes of 1+ hour	Max detected hashes
No limit	**2,022**	**524**	**3,684**	**59**	**23**
10 MAC	2,001	524	3,772	59	30
5 MAC	1,958	524	2,192	62	41
1 MAC	1,847	524	1,776	61	233

Overall, one can observe that limiting the number of MAC addresses has a harmful impact on the privacy (cf. Section 4.1), but also on the accuracy of IndoorHash. Indeed, the variant considering a single MAC address offers the worst performances: the maximum number of occurrences drops to 1,776 compared to the default variant that captures up to 3,684 occurrences. This can result in some physical places not being recognized by the algorithm. At the same time, one can see that most important places (*Hashes with 1+ hour*) keep being recognized with the same accuracy as the baseline, thus indicating that limiting MAC addresses tends to reduce the capability to capture places visited for shorter periods.

5 Towards Privacy-preserving Location-based Services

In this paper, we believe that *Location-Based Services* (LBS) require to guarantee the user privacy by design, in order to avoid any potential risk of data leaks that might contribute to learn sensitive knowledge from visited locations (*e.g.*, home, office, leisure). Such privacy-preserving LBS cannot collect and process the raw user locations, but should rather build on location hashes that reveal the necessary bits of information that are required to implement the service. This section therefore lists candidate LBS features that could benefit from IndoorHash in the future.

Points of Interest (POIs) refer to known physical locations that may reveal sensitive information about the habits and tastes of end users (*e.g.*, shopping mall, restaurants) and can be used by third parties to feed recommendation

algorithms. IndoorHash can be considered as fine-grained POI that does not reveal any semantics about the location visited by a user, but can still be used to recommend alternative nearby locations by comparing similar hash histories among users.

Social networking is another example of LBS that can benefit from IndoorHash by comparing the hash histories in order to identify potential connections within a crowd of users without revealing the exact locations where these users use to meet. IndoorHash can therefore support the definition of proximity datasets that are commonly used for various purposes, such as mobile testing [22], by reporting on the colocation of end users and potential connections in a crowd that can support the evaluation of dissemination protocols [21, 19].

Context prediction can also leverage the history of IndoorHash to understand the mobility of end users and possibly predict the next location (*e.g.*, using markov chains or n-grams) of a given user in order to anticipate some actions, like fetching some content or buffering a video before loosing an Internet connection [25]. This context prediction can also be used to support the development of context-aware applications whose behaviors can adjust accordingly. For example, maintaining a different profile of a web browser to avoid that a private or professional web history being exploited by third-party cookies in a different context, which might be unknown and thus sensitive.

Routine analysis refers to the detection of user activities to better understand her daily habits. While DayKeeper (cf. Section 4.2) provides an example of personal activity tracker built on top of IndoorHash, we believe that one can provide advanced features to detect if a user is in or out of a routine and react accordingly. Such a mechanism can help the user to focus on her tasks (*e.g.*, by enabling a *do not disturb* mode) or trigger specific assistance when she is not in a routine.

Mobile crowdsourcing consists in gathering field measurements from a crowd of participants. In mobile crowdsourcing systems, the user location is often used to group data along spatial dimensions to build some specific maps (*e.g.*, the open signal initiative)[6]. In such cases, beyond raw GPS locations, Geohash and Pluscode[7] provide a compact encoding of any physical location on earth. While both of them can fuzz the user location by adjusting their encoding scheme, they keep disclosing a critical information for the end user. Yet, user location may only be required to group or aggregate field measurements by location without sharing the physical location with anyone. In these situation, we believe that IndoorHash can provide a privacy-preserving spatial keys that can be used to process geolocated measurements. Interestingly, the indexing structure allows end users who can compute the IndoorHash to access these aggregated

[6] https://www.opensignal.com/
[7] https://plus.codes/

measurements. For example, one can imagine a user getting the history of indoor air pollution using her current hash to query a remote measurement service built on top of IndoorHash [11].

6 Conclusion

Location-Based Services (LBS) are increasingly getting adopted by end users who expect to be delivered personalized user experiences depending on their current context. However, the characterization of this context often relies on the physical location of the user which is *i)* hard to capture in indoor environments and *ii)* highly sensitive from a privacy perspective.

In this paper, we tackle both of these challenges by introducing a new indoor location hash, named IndoorHash. More specifically, IndoorHash captures a logical user location without disclosing the physical place visited by this user. Our contribution consists in applying the SimHash algorithm to the processing of MAC addresses exposed by WiFi APs. We show that the IndoorHash computed from such input data offers a robust location hash that accurately capture any user location. We also implement a distributed system that allows to store and index IndoorHash in order to ease the detection of known locations and share geolocated data with privacy guarantees.

As a matter of perspectives, we are interested in extending the IndoorHash to a more general location hash scheme that can leverage additional signals surrounding a user, like the GSM antennas or BLE beacons in order to increase the coverage of inferred locations. We are also interested in exploring new operations on location hashes to support privacy-preserving data clustering algorithms.

References

1. Ahmed, A.U., Bergmann, N.W., Arablouei, R., Kusy, B., De Hoog, F., Jurdak, R.: Poster Abstract: Fast Indoor Localization Using WiFi Channel State Information. In: 17th Int. Conf. on Information Processing in Sensor Networks (IPSN'19). pp. 120–121. IEEE (apr 2018)
2. Beder, C., Klepal, M.: Fingerprinting based localisation revisited: A rigorous approach for comparing rssi measurements coping with missed access points and differing antenna attenuations. In: Int. Conf. on indoor positioning and indoor navigation (IPIN). pp. 1–7. IEEE (2012)
3. Birant, D., Kut, A.: ST-DBSCAN: An algorithm for clustering spatial-temporal data. Data & Knowledge Engineering **60**(1) (2007)
4. Capurso, N., Song, T., Cheng, W., Yu, J., Cheng, X.: An Android-Based Mechanism for Energy Efficient Localization Depending on Indoor/Outdoor Context. IEEE Internet of Things Journal **4**(2), 299–307 (apr 2017)
5. Caso, G., De Nardis, L., Lemic, F., Handziski, V., Wolisz, A., Di Benedetto, M.G.: Vifi: Virtual fingerprinting wifi-based indoor positioning via multi-wall multi-floor propagation model. IEEE Transactions on Mobile Computing (2019)
6. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: 34th ACM Symp. on Theory of computing. pp. 380–388. ACM (2002)

7. Choi, T., Chon, Y., Cha, H.: Energy-efficient WiFi scanning for localization. Pervasive and Mobile Computing **37**, 124–138 (jun 2017)
8. De Montjoye, Y.A., Hidalgo, C.A., Verleysen, M., Blondel, V.D.: Unique in the crowd: The privacy bounds of human mobility. Scientific reports **3**, 1376 (2013)
9. Del Corte-Valiente, A., Gómez-Pulido, J.M., Gutiérrez-Blanco, O.: Efficient techniques and algorithms for improving indoor localization precision on wlan networks applications. Int. Journal of Communications, Network and System Sciences **2**(07), 645 (2009)
10. Haderer, N., Rouvoy, R., Seinturier, L.: Dynamic Deployment of Sensing Experiments in the Wild Using Smartphones. In: 13th Int. Conf. on Distributed Applications and Interoperable Systems (DAIS'13). LNCS, vol. 7891, pp. 43–56. Springer (2013)
11. Hanoune, B., Kassi, R., Verbeke, B., Assy, E., Clavier, L., Crumeyrolle, S., Degrande, S., Le Pallec, X., Rouvoy, R.: Conception and deployment of the Apolline sensor network for IAQ monitoring. In: 10th Int. Conf. on Indoor Air Quality, Ventilation and Energy Conservation in Buildings (IAQVEC'19) (Sep 2019)
12. He, S., Chan, S.H.G.: Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons. IEEE Communications Surveys & Tutorials **18**(1), 466–490 (2015)
13. Jin, M., Koo, B., Lee, S., Park, C., Lee, M.J., Kim, S.: IMU-assisted nearest neighbor selection for real-time WiFi fingerprinting positioning. In: Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN'14). pp. 745–748. IEEE (oct 2014)
14. Krumm, J., Horvitz, E.: Locadio: Inferring motion and location from wi-fi signal strengths. In: mobiquitous. pp. 4–13 (2004)
15. Li, H., Sun, L., Zhu, H., Lu, X., Cheng, X.: Achieving privacy preservation in WiFi fingerprint-based localization. In: IEEE INFOCOM. pp. 2337–2345. IEEE (apr 2014)
16. Li, Z., Zhao, X., Hu, F., Zhao, Z., Carrera, J.L., Braun, T.: Soicp: A seamless outdoor-indoor crowdsensing positioning system. IEEE internet of things journal (2019)
17. Liu, H., Gan, Y., Yang, J., Sidhom, S., Wang, Y., Chen, Y., Ye, F.: Push the limit of WiFi based localization for smartphones. In: 18th Int. Conf. on Mobile computing and networking (Mobicom'12). p. 305. ACM, New York, New York, USA (2012)
18. Luo, C., Hong, H., Chan, M.C.: Piloc: A self-calibrating participatory indoor localization system. In: 13th Int. Symp. on Information Processing in Sensor Networks (IPSN'14). pp. 143–153. IEEE (2014)
19. Luxey, A., Bromberg, Y.D., Costa, F.M., Lima, V., da Rocha, R.C., Taïani, F.: Sprinkler: A probabilistic dissemination protocol to provide fluid user interaction in multi-device ecosystems. In: Int. Conf. on Pervasive Computing and Communications (PerCom). pp. 1–10. IEEE (2018)
20. Lymberopoulos, D., Liu, J.: The microsoft indoor localization competition: Experiences and lessons learned. IEEE Signal Processing Magazine **34**(5), 125–140 (2017)
21. Meftah, L., Rouvoy, R., Chrisment, I.: Fougere: User-centric location privacy in mobile crowdsourcing apps. In: Int. Conf. on Distributed Applications and Interoperable Systems (DAIS'19). pp. 116–132. Springer (2019)
22. Meftah, L., Rouvoy, R., Chrisment, I.: Testing nearby peer-to-peer mobile apps at large. In: 6th Int. Conf. on Mobile Software Engineering and Systems (MOBILESoft'19). pp. 1–11. IEEE (2019)

23. de Montjoye, Y.A., Quoidbach, J., Robic, F., Pentland, A.S.: Predicting personality using novel mobile phone-based metrics. In: Int. Conf. on social computing, behavioral-cultural modeling, and prediction. pp. 48–55. Springer (2013)
24. Nguyen, T.B., Nguyen, T., Luo, W., Venkatesh, S., Phung, D.: Unsupervised inference of significant locations from wifi data for understanding human dynamics. In: 13th Int. Conf. on Mobile and Ubiquitous Multimedia. pp. 232–235. MUM '14, ACM, New York, NY, USA (2014)
25. Paspallis, N., Alshaal, S.E.: Improving qoe via context prediction: A case study of using wifi radiomaps to predict network disconnection. In: ICPE Companion. pp. 31–34. ACM (2017)
26. Pulkkinen, T., Verwijnen, J., Nurmi, P.: WiFi positioning with propagation-based calibration. In: 14th Int. Conf. on Information Processing in Sensor Networks (IPSN'15). pp. 366–367. ACM (2015)
27. Rekimoto, J., Miyaki, T., Ishizawa, T.: Lifetag: Wifi-based continuous location logging for life pattern analysis. In: LoCA. vol. 2007, pp. 35–49 (2007)
28. Sakib, M.N., Halim, J.B., Huang, C.T.: Determining location and movement pattern using anonymized WiFi access point BSSID. In: 7th Int. Conf. on Security Technology (SecTech'14). pp. 11–14. IEEE (dec 2015)
29. Salamah, A.H., Tamazin, M., Sharkas, M.A., Khedr, M.: An enhanced WiFi indoor localization System based on machine learning. In: 2016 Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN'16). pp. 1–8. IEEE (oct 2016)
30. Sapiezynski, P., Stopczynski, A., Gatej, R., Lehmann, S.: Tracking human mobility using WiFi signals. PLoS ONE **10**(7) (2015)
31. Sapiezynski, P., Stopczynski, A., Wind, D.K., Leskovec, J., Lehmann, S.: Offline behaviors of online friends. arXiv preprint arXiv:1811.03153 (2018)
32. Shen, G., Chen, Z., Zhang, P., Moscibroda, T., Zhang, Y.: Walkie-markie: Indoor pathway mapping made easy. In: 10th USENIX Symp. on Networked Systems Design and Implementation (NSDI'13). pp. 85–98 (2013)
33. Stopczynski, A., Sekara, V., Sapiezynski, P., Cuttone, A., Madsen, M.M., Larsen, J.E., Lehmann, S.: Measuring large-scale social networks with high resolution. PloS one **9**(4), e95978 (2014)
34. Wind, D.K., Sapiezynski, P., Furman, M.A., Lehmann, S.: Inferring stop-locations from WiFi. PLoS ONE **11**(2) (2016)
35. Xie, X., Xu, H., Yang, G., Mao, Z.H., Jia, W., Sun, M.: Reuse of WiFi information for indoor monitoring of the elderly. In: 17th Int. Conf. on Information Reuse and Integration (IRI'16). pp. 261–264. IEEE (jul 2016)
36. Yiu, S., Dashti, M., Claussen, H., Perez-Cruz, F.: Wireless rssi fingerprinting localization (2017)
37. Yiu, S., Yang, K.: Gaussian Process Assisted Fingerprinting Localization. IEEE Internet of Things Journal **3**(5), 683–690 (oct 2016)
38. Zandbergen, P.A.: Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning. Transactions in GIS **13**, 5–25 (2009)
39. Zhang, H., Yan, Z., Yang, J., Tapia, E.M., Crandall, D.J.: Mfingerprint: Privacy-preserving user modeling with multimodal mobile device footprints. In: Int. Conf. on Social Computing, Behavioral-Cultural Modeling, and Prediction. pp. 195–203. Springer (2014)
40. Zhang, N., Feng, J.: Polaris: A fingerprint-based localization system over wireless networks. In: Int. Conf. on Web-Age Information Management. pp. 58–70. Springer (2012)