Challenges in Automated Measurement of Pedestrian Dynamics

Maarten van Steen[1(✉)], Valeriu-Daniel Stanciu[2], Nadia Shafaeipour[3], Cristian Chilipirea[4], Ciprian Dobre[5], Andreas Peter[6], and Mingshu Wang[7]

[1] Digital Society Institute, University of Twente, Enschede, The Netherlands
m.r.vansteen@utwente.nl
[2] Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands
[3] Faculty of Geo-information Science and Earth Observation, University of Twente, Enschede, The Netherlands
[4] Azure Cloud, Microsoft, Bucharest, Romania
[5] Faculty of Automatic Control and Computer Science, Politehnica University of Bucharest, Bucharest, Romania
[6] Department of Computer Science, University of Oldenburg, Oldenburg, Germany
[7] School of Geographical & Earth Sciences, University of Glasgow, Glassgow, UK

Abstract. Analyzing pedestrian dynamics has since long been an active and practical field of interest. Since the introduction of, in particular, smartphones, various organizations saw a simple means for automatically measuring pedestrian dynamics. The basic idea is simple: network packets sent by WiFi-enabled devices can be collected by sensors and by extracting the unique MAC address from each packet, it should be possible to count how many devices are detected by a single sensor, as well as how devices move between sensors. Although this approach has been commercially deployed for many years, it is now largely forbidden (at least in the EU) due to obvious privacy infringements. In this paper, we address challenges and some potential solutions to automated measurement of pedestrian movements while protecting privacy. The results come from learning the hard way: having run experiments extensively over the past years, we have gradually gained considerable insight in what is possible and what may lie ahead.

1 Introduction

Understanding pedestrian dynamics is a long-standing scientific field motivated by questions from very different domains (e.g., tourism [11], urban planning [14], safety and security [12]). Automating the measurement of pedestrian dynamics allows for collecting more accurate data than what is possible by manual means. In the past decade, much attention has been spent on using the fact that people carry network-connected devices such as smartphones (see, e.g., the extensive surveys conducted by

This is an internally reviewed accompanying paper to a keynote delivered by Maarten van Steen at DAIS 2022, and is to be considered as background information for that talk.

Published by Springer Nature Switzerland AG 2022
D. Eyers and S. Voulgaris (Eds.): DAIS 2022, LNCS 13272, pp. 187–199, 2022.
https://doi.org/10.1007/978-3-031-16092-9_12

our team [7,23]). Such devices regularly transmit network packets, and many of those packets contain information that uniquely identifies the transmitter, such as its MAC address in WiFi or Bluetooth communications. The basic idea is that such identifiers can be used as a proxy for the person carrying the associated device. In this way, it becomes possible to, in principle, gather statistics on the whereabouts of a pedestrian by simply capturing his or her movements through identification of the device that is being carried.

There are several problems with this approach. For one, such schemes infringe upon a person's privacy and are largely forbidden, as formulated in the European General Data Protection Regulation (GDPR). However, the strict regulation on the automated collection of data transmitted by devices is being alleviated in the case data is used for **statistical counting**, and under the condition that pedestrians are informed, as well as that the data is discarded after the statistics have been computed [6]. Although these measures allow for some automated gathering of data, the question of how to do so in a privacy-preserving manner remains open.

Next to privacy infringements, there are other problems pertaining to the data gathered by collecting network packets. Many modern devices use randomized MAC addresses whenever possible, effectively making it impossible to check whether a device has been detected for a long time at a specific location, or has moved between two locations. MAC address randomization is offered by device manufacturers in light of privacy considerations, but it is not a technique that can be applied to all packets. Likewise, detecting packets in outdoor environments is by itself already difficult, certainly if packets can be captured by multiple sensors at the same time as it makes it much harder to determine the location of a device.

In this paper, written for a nonexpert, we address several of these problems, with an emphasis on privacy protection. We discuss potential solutions based on our own experience with experiments we conducted in the past five years, as well as some solutions that are currently being explored by others.

2 Automated Measurement of Pedestrian Behavior

Key to automated measurement of pedestrian dynamics is capturing network traffic from a device carried by a person and extracting an identifier of that device. Capturing network traffic and extracting an identifier is done by means of a **sensor** that knows the communication protocol (WiFi, Bluetooth). A WiFi access point can be re-purposed to record the MAC network address transmitted by a device and use these as device identifiers. However, such a **raw device identifier** is considered to be personal information and recording it can be considered intrusive. Common practice has therefore been to transform a raw device identifier *RID* to a **pseudonym** *PID* using a secure one-way encryption function F (e.g., a collision-resistant cryptographic hash function). The pseudonym is unique for a given raw device identifier, but the function makes it computationally infeasible to derive the raw identifier from the pseudonym. In other words, the inverse function F^{-1} is infeasible to compute. In this way, one can use the pseudonym $PID = F(RID)$ as a device identifier. Given a sufficiently strong hash function (several of which are known to exist, such as SHA256), it is impractical to determine the *RID* from the *PID* alone.

Schemes are also being deployed in which a provably secure **parameterized** one-way encryption function is used to generate the pseudonym. To illustrate, consider such a function $F_{T,S}$ with two parameters, a parameter T related to a **time span**, and a parameter S associated with a group of one or more specific **sensing locations**. The values of the parameters determine which pseudonym is generated, given an identifier as input. In other words, if we take *RID* as the raw device identifier, we can recognize that device through the pseudonym $PID = F_{T,S}(RID)$ (here T and S can be considered what are known as salts for the hash function). If we change either T or S, $F_{T,S}(RID)$ will change as well. Note that F is collision resistant: when changing the value for T or S, a pseudonym is generated that makes it practically impossible to associate it with any other pseudonym based on *RID* [9].

To illustrate, if T spans only a single day, yet S is the same for a number of sensing locations, it becomes computationally impossible to identify the same physical device over periods lasting more than one day. It will be possible to identify the same device as being at different sensing locations during a single day. If we change S per location (i.e., different sensors use different encryption functions), identifying the movement of a device across multiple sensing locations is computationally infeasible.

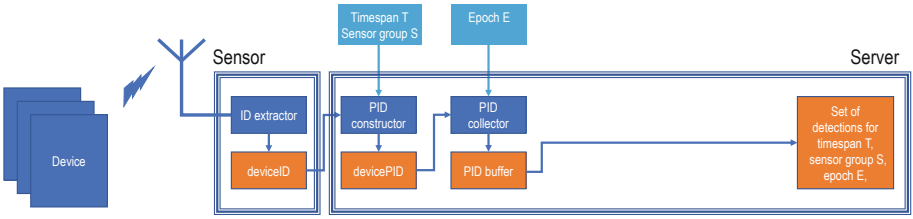

Fig. 1. The general organization of many current sensors for measuring pedestrian dynamics.

Current approaches are summarized in Fig. 1. Devices transmit signals that are captured by a sensor σ, which subsequently extracts a *device identifier*, typically the aforementioned *RID*. This identifier is then (securely) transmitted to a server where it is converted to a *device pseudonym*, and handed over to a *PID collector*. The latter generally collects *PIDs* for a relatively small detection period, or time window, referred to as an **epoch**, and places detected *PIDs* in a buffer, ignoring *PIDs* that had already been encountered during that epoch. An epoch typically lasts 5 min and effectively identifies the time of a detection (e.g., as the tuple $\langle year, month, day, hour, timeslot \rangle$). In this way, the collector registers the unique devices it has seen during that epoch and avoids any double counting. At the end of the epoch, each collected device is added to the set of detections as a tuple $\langle PID, E, \sigma \rangle$, which tells when ($E$) and where (σ) a device with identifier *PID* was detected.

In practical situations, we see that commercial companies use so-called WiFi probe messages to identify devices, let T span a single day, and often use a different value for S per sensor. A probe message is broadcast by a WiFi-enabled device in search of a WiFi access point to set up a connection. Such schemes allow for footfall counting on

a per-daily basis. We also see cases where T spans much longer intervals and where S is the same for all sensors, which effectively allows for tracking. Recently, this led to a fine of 600k € in The Netherlands for violation of the GDPR.[1]

In the following, we assume sensor nodes to be trusted and to follow a single fundamental design principle:

> **FDP1**: Any data produced by a collection of (trusted) sensors cannot be traced back to a physical device.

This principle states that no matter how we combine the data coming from sensors, it should be impossible to identify an actual physical device, and thus its owner. As a consequence, any system processing this data, data generated by any collection of trusted sensors, is secure by design. Ideally, no extra security measures need to be implemented for the processing system. Note that stating the principle does not mean that it can be easily established. For example, when only very few devices are detected, it may become difficult to protect the privacy of their owners. Also, as we shall see, we do need to rely on a noncolluding server.

Given the GDPR and its alleviation for measuring pedestrian dynamics, we focus our work on **secure and privacy-aware statistical counting**. In particular, we want a system that can address the following two types of queries:

- **Query type 1**: How many people have been at location L during time span T?
- **Query type 2**: How many people, when at location $L1$ during time span $T1$, were at location $L2$ during time span $T2$?

Given that not everyone will carry a network-enabled device, and that some people may also have more than one such device, it is clear that precise counting is out of the question. Moreover, many modern smartphones now deliberately use nonidentifiable information when transmitting specific network packets (namely for the WiFi probe requests which are used for automated measurements), typically in the form of randomized MAC addresses. In addition, it is well known that radio signals as used in wireless communication systems often exhibit highly unpredictable behavior [4]. This means that we need to deal with missing network packets, but also packets which are received at a much larger distance than would normally be possible considering the specifications of the wireless medium.

In addition, simply assuming that sensors can be trusted is easier said than done. We therefore also require the following:

> **FDP2**: A sensor may not store any information that may be traced back to a physical device any longer than strictly necessary for statistical counting,

> **FDP3**: A sensor may not share information that may be traced back to a physical device.

In other words, even storing pseudonyms needs to be limited to a minimal amount of time and those pseudonyms need to be confined to the sensor. This leads to the design sketched in Fig. 2.

[1] https://autoriteitpersoonsgegevens.nl/en/news/dutch-dpa-fines-municipality-wi-fi-tracking.

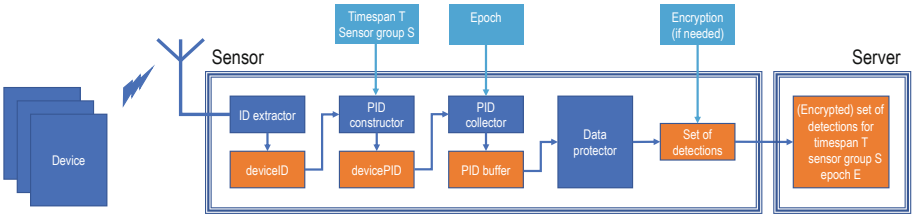

Fig. 2. The design of a privacy-protecting sensor.

There are a number of important differences from current approaches. First and foremost, we introduce a separate **data protector**. A data protector takes a collection of *PIDs* as input and produces a set of detections from which it is computationally infeasible to extract any original *PID*. We explain below how this can be achieved. Second, the whole path from ID extraction to an anonymized set of detections takes place at the (trusted) sensor. We do not trust the server collecting data from many sensors or the communication medium between the sensor and the server. If we need to store data external to the sensor, we do so only after the sensor has taken sufficient privacy-protecting measures. Thirdly, all data that we stored until handed over to the protector is deleted after the elapse of an epoch. Only the protected set of detections is kept.

3 Protecting Privacy Through Detection k-anonymity

3.1 Approach

In a first attempt to protect privacy, we developed a method based on achieving k-anonymity [15]. To explain, let **PID** denote the set of all possible *PIDs*. We devise a mapping m to a new set of pseudonyms **MPID**, such that for each detected $pid \in$ **PID** there are at least $k-1$ other detected pseudonyms $\{pid_1, \ldots, pid_{k-1}\} \subset$ **MPID** with $m(pid) = m(pid_i)$. To rephrase this, assuming that each sensor stores mapped pseudonyms, then for each such stored pseudonym associated with some epoch, we are guaranteed that the sensor actually detected at least k different devices during that epoch. We denote such a mapped pseudonym as a **multipseudonym**. It is important that combining multipseudonyms preserves this k-anonymity.

A straightforward mapping is the one that simply truncates detected pseudonyms. This works fine, in particular if we can assume that pseudonyms are effectively drawn uniformly at random (which can be achieved by using a secure hashing function that generates a pseudonym from a detected MAC address). A uniform distribution guarantees that no biases are introduced when removing bits, and thus no systematic error when trying to map multiple *PIDs* to the same multipseudonym. For each epoch, a sensor then stores truncated pseudonyms as multipseudonyms and records how many different pseudonyms it detected for each stored multipseudonym. A problem, however, is that we need to determine *a priori*, i.e., at design time, how many bits to keep without knowing if that choice will lead to having detected enough pseudonyms to guarantee

k-anonymity. In other words, truncation of detected pseudonyms may leave us with multipseudonyms for which there are simply less than k detected pseudonyms. In that case, we have no choice than to discard those multipseudonyms (and thus also the counts of the number of associated detected pseudonyms). Clearly, this may seriously affect the accuracy of counting pedestrians.

As an alternative to discarding multipseudonyms, we can also remap detected pseudonyms such that k-anonymity is preserved. To this end, we need to consider only the multipseudonyms and their associated detected pseudonyms that violate k-anonymity. A naive approach is to generate an unused multipseudonym and assign that to the first k detected pseudonyms (for which k-anonymity was violated); generate another unused multipseudonym for the next k detected pseudonyms, and so on. This solution works fine for a single location, but not if we want to count how many pedestrians moved from location A to B. The problem is that any relation with what was detected at A is lost when generating multipseudonyms at B (at least for those detected pseudonyms that violated k-anonymity). We then might have just as well discarded them.

We thus need a *systematic* way of mapping k-anonymity-violating detected pseudonyms (we refer to them simply as violating pseudonyms), and apply that method to all sensors. We proceed as follows with what we denote as a **correction method**. Assume a sensor has n violating pseudonyms for a specific epoch. It then sorts those n pseudonyms and subsequently keeps only the top $\lfloor n/k \rfloor$ ones. Using the remaining $n - \lfloor n/k \rfloor$ violating pseudonyms, it then systematically increases the counts for every one of the remaining top of violating pseudonyms. In this way, in principle, almost none of the counts for the other violating pseudonyms are lost. Moreover, if this procedure is used at location A as well as B, we see that both locations will be assigning the same multipseudonym to the same detected pseudonym, just as we wanted. The details can be found in [15], along with a proof that when results from different sensors are combined, k-anonymity is preserved.

3.2 Evaluation

We have evaluated this setup using simulations as well as real-world data. For the latter, we used data on subway trips from Beijing [22,24]. That data set can be used to mimic WiFi-based detections. The set consists of check-in and check-out records, each record containing a unique card identifier, the identifier of the station where the card is being checked, as well as a timestamp. For our purposes, namely counting the number of devices that were detected at location A during some epoch e_1 and later at location B during epoch e_2, the data is just fine: each check-in or check-out corresponds to a WiFi-based detection of a device; the card identifier is analogous to a MAC address. (Note that although we also have real-world measurements on WiFi data, those measurements do not provide us with ground truth: they do not tell us which device actually moved from one location to another. In contrast, the Beijing data set gives us an accurate account of movements, making it, in principle, ideal for mimicking WiFi-based measurements.)

We apply the k-anonymity algorithm as explained above, for different values of the epoch length, the size of truncated pseudonyms (i.e., the number of bits to keep), as well as for different values of k. A check-in or check-out counter is treated as a sensor. The idea is that each counter applies the algorithm and sends the k-anonymized data to

a central server. If we consider an isolated trajectory, i.e., only those trips that have been made between two specific locations, we attain high accuracy of counting the number of trips between two locations. Results are generally better for lower values of k, yet this is partly explained by the sometimes limited number of actual detections during an epoch. We also see that there is a trade-off between the length of an epoch and accuracy: the smaller an epoch is, the fewer detections we will have, in turn affecting the accuracy (depending on k). The length of an epoch becomes less important once enough detections can be guaranteed.

However, matters may easily deteriorate when combining trips, as also examined in [15]. Let us return to the situation of counting pedestrians moving from A to B. Assume that the sensor at A collected a set of pseudonyms $\mathbf{PID_A}$, which were then mapped to the multiset $\mathbf{MPID_A}$. Likewise, at location B we have sets $\mathbf{PID_B}$ and $\mathbf{MPID_B}$. If there are no intermediate junctions, the multiset $\mathbf{MPID_A} \cap \mathbf{MPID_B}$ represents the devices that had moved from A to B.

Now consider the situation that we have two intermediate junctions $Z1$ and $Z2$ on the path from A to B. At $Z1$ the sensor detects $\mathbf{PID_{Z1}}$ devices entering the flow of pedestrians moving from A to B, which are mapped to the multiset $\mathbf{MPID_{Z1}}$. Assume that $\mathbf{MPID_A} \cap \mathbf{MPID_{Z1}}$ is nonempty. At junction $Z2$, the sensor detects $\mathbf{PID_{Z2}}$ devices (mapped to $\mathbf{MPID_{Z2}}$) *leaving* the flow again. If $\mathbf{PID_{Z2}} \subseteq \mathbf{PID_A}$ then, clearly, the final count at B will be false: it will have been contaminated by devices entering at $Z1$ that were mapped to the same multipseudonyms as those at A, which then for counting purposes go unnoticed by the devices that left at $Z2$. This situation is sketched in Fig. 3.

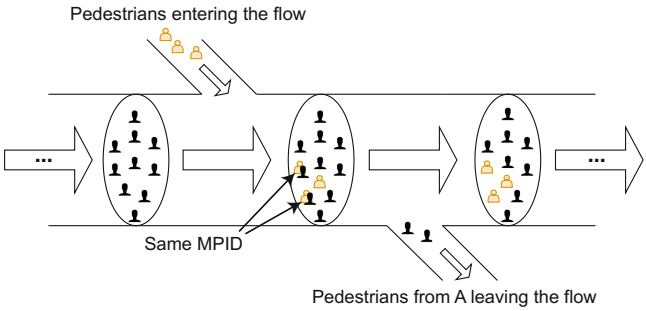

Fig. 3. The effect of devices entering and leaving a flow.

3.3 Reflection

Although the described approach toward k-anonymity is highly efficient, it turns out that it is fairly sensitive to perturbations in flows of pedestrians between two locations. Moreover, when considering practical situations such as subway networks, we have found that setting correct values for epoch length and truncation sizes may be tricky. It is yet unclear whether the approach is practically feasible.

4 Protecting Privacy Through Homomorphically Encrypted Bloom Filters

4.1 Approach

An alternative to k-anonymity is to have sets of pseudonyms represented by **Bloom filters** [2]. A Bloom filter is a constant-space probabilistic storage mechanism. Being probabilistic, a Bloom filter can indicate that an item that was not included in the set is actually present. The opposite is not true, if the Bloom filter indicates that an item is not in the set it is guaranteed that this is true. The Bloom filter is represented using a vector of m bits, initially all set to 0. Using a collection of k hash functions, an element x is added to a Bloom filter by setting the position $h_i(x)$ to 1 for each of the k hash functions. Each element can thus be represented as an m-bit vector consisting of exactly k bits set to 1. A bitwise OR operation is performed each time an element is added.

A Bloom filter has the important property that one cannot retrieve the elements of the set it represents other than by exhaustively testing for all possible elements. In other words, a Bloom filter supports *only* membership tests. To check if x is in a set A, one needs to check if every position $h_i(x)$ for the Bloom filter BF representing A has been set to 1:

$$x \in A \text{ only if } \prod_{i=i}^{k} BF[h_i(x)] = 1$$

(In the following, we will use the same notation for a set A and its representation by means of a Bloom filter.) A Bloom filter allows for testing whether an element is in the union of two sets, or in their intersection (this can be done by applying the OR operation for union and the AND operation for intersection). This forms the basis for counting at a single location (perhaps using multiple sensors), or counting movements (between different locations, and certainly using multiple sensors). Statistical counting is possible by means of a simple estimation n^* of the number of elements in a Bloom filter [17].

$$n^* = -\frac{m}{k} \ln\left(1 - \frac{X}{m}\right)$$

where X is the number of nonzero elements in the Bloom filter.

One problem with the approach sketched so far is that the sensors would, in principle, need to share Bloom filters, which violates our design principle *FDP3*. Considering that testing for membership entails a bitwise AND operation, which is equivalent to a multiplication of 0's and 1's, we can test for such membership using **multiplicative homomorphic encryption**. Such an encryption scheme enables the multiplication of two encrypted numbers without the need to first decrypt those numbers. To clarify, let $[p]$ denote the homomorphically encrypted version of the number p. Then, with multiplicative homomorphic encryption, we have

$$[p] * [q] = [p * q]$$

The basic idea is that a third party (e.g., a crowd expert) who needs the value of a statistical count provides a **public key** by which each sensor homomorphically encrypts

the entries of its Bloom filter. Note also that if a value p is encrypted, leading to $[p]_1$, and that same value is encrypted a next time, leading to $[p]_2$, the two encrypted values will be different: $[p]_1 \neq [p]_2$ and an observer will not be able to distinguish the two underlying values to be the same. Such an encrypted filter can be handed out to another sensor without disclosing any detections. This latter sensor can still compute an intersection $[A]$ using its own Bloom filter (which has been homomorphically encrypted with the same public key). Then, by simply shuffling the elements of that (encrypted) intersection, the result will be an encrypted Bloom filter $[A^*]$, in principle representing a *different* set of devices (which have nothing to do with the actual detected devices) yet of the same estimated size: $A \neq A^*, |A| = |A^*|$. This encrypted Bloom filter $[A^*]$ is handed out to the entity having the private key, who can then compute $|A^*|$. Obviously, no sensor would ever hand out its (encrypted) Bloom filter to the third party, as this would violate *FDP3*.

4.2 Evaluation

Our initial motivation for developing an anonymization technique based on k-anonymity was our assumption that using Bloom filters was simply too expensive in terms of computational and storage resources. At that time, we were considering that sensors would need to do handle all possible queries and also store results for those queries. We were wrong. For many practical situations, using a Raspberry Pi4 as the basis for a sensor, in combination with offloading the encrypted Bloom filters to a centralized server, is enough. In our experiments, assuming that a sensor needs to detect at most 10,000 *PIDs* during a single epoch (again, meaning to be able to detect at most 10,000 different devices during, say, 5 min), it takes just over 2 min for a sensor to process a complete pipeline of collecting data, constructing a Bloom filter, and subsequently encrypting the filter. The implementation is optimized in the sense that it makes optimal use of multiple cores. A serial implementation takes close to 8 min. When we can assume that at most 1000 devices need to be detected per epoch, these numbers drop to tens of seconds. Not surprisingly, the server, even lightweight versions, can easily handle the generated workloads. Our conclusion is that there are no serious problems when it comes to performance.

We have also tested our setup against real data, gathered during a multi-day outdoor festival. In this case, we evaluated how our method of privacy protection would lead to the same results as the ones coming from the dataset as collected by the sensors that were using the original pseudonyms. Because Bloom filters are probabilistic in nature, deviations are to be expected in comparison to processing raw data. Again, we see that we attain high accuracies in the 90–98% ranges for both footfall counting as well as measuring the size of crowd flows. Some of these results have been reported in [16].

4.3 Reflection

There are reasons to believe that the described Bloom-filter approach is the way to go for automatically measuring pedestrian dynamics. Yet, there are several challenges that need to be addressed before drawing final conclusions.

The setup described so far implicitly relies on the assumption that devices are detected by only one sensor at a time (this is why we can state that a flow moved from one location to another). In practice, avoiding simultaneous detections of the same device by multiple sensors may not be possible, or even desirable. For example, for purposes of reliability, we may wish to install multiple sensors at a single location and combine their detections as if they came from one, more powerful, sensor. In principle, this is possible by constructing the union of (encrypted) Bloom filters over the same epoch but from different sensors. Such a construction can be efficiently done by a server, as described above. To what extent unions affect the design of Bloom filters remains to be seen: there is a trade-off between the length m, the number of hash functions k, and the accurate representation of sets of a given size, although unions of Bloom filters are known to be lossless. More important is that constructing unions requires **additive homomorphic encryption**, implying that we may need a more advanced encryption scheme, or use two partial homomorphic schemes side-by-side. It is yet unclear what this would mean for the design and implementation of the monitoring system as a whole. Homomorphic encryption schemes are known to be generally computationally hungry [1].

A final remark is in place. We essentially looked at combining only two Bloom filters, and relied on the closed formula for estimating the size of the intersection [17], as well as an improvement for that formula [13]. However, Bloom filters are probabilistic data structures, meaning that when more than two filters are combined it is seen that the estimated size becomes increasingly less accurate. No closed formula is known for combining more than two Bloom filters. This implies that for practical implementations, we need to look much closer into the accuracy of the final result after having combined, in whatever way, several Bloom filters.

5 Other Challenges

Although protecting privacy has been a major issue in automated measuring pedestrian dynamics, there are many other issues that need to be taken into account. Let us consider a number of those that we encountered in the past years, of which some have also been reported in [4].

5.1 Behavior of Carry-On Devices and (non)overlapping Sensor Ranges

The whole idea of automated measurements assumes that carry-on devices send out packets at some minimal frequency. We have found that this may be a flawed assumption. In fact, the behavior of different devices, especially from different manufacturers, may vary widely, as also reported by others [10]. Together with the fact that wireless communication is inherently difficult, making many transmitted packets impossible to detect, even when in advertised range, means that automated measurements have an unexpectedly low number of detections. One possible solution is to increase the length of an epoch, as it simply increases the chance of capturing packets from a device that is within range of a sensor. On the other hand, long epochs can easily complicate deciding where a device actually is: just imagine that within a single epoch a device is detected

at two locations (which may easily happen when a device moves from one location to another within that epoch). Effectively, increasing epochs means that devices may more easily be detected by multiple sensors during the same epoch. Yet, even if epochs are small enough, we still need to handle the case in which sensor ranges overlap. Determining a location of a detected device may turn out to be difficult, in turn, hindering the process of determining the size of crowd flows.

After running real-world experiments trying to distinguish bystanders looking at a marching crowd and the marching crowd itself, Groba [10] draws the conclusion that the behavior of carry-on devices may be so unpredictable that it may be close to impossible to answer this type of questions or to accurately measure pedestrian dynamics.

5.2 MAC-Address Randomization

The statistical counting techniques described above are independent of the actual technique for detecting a device. So far, we have gained considerable experience with WiFi-based detections, which have shown to be prone to many unreliable measurements, notably in outdoor environments [3,5]. Making things more difficult is the generalization of the use of MAC-address randomization. When devices are associated to an access point, such as, for example, when roaming within a public network or within international networks such as Eduroam, they will always use their uniquely assigned MAC address. In other cases, when a device is actively seeking for a network, we see that increasingly often MAC-address randomization is deployed. This randomization makes it much more difficult to identify devices. Attempts have been made to automatically fingerprint devices by considering other fields in the transmitted signals (see, e.g., [20], which follows a machine-learning approach). We are investigating to what extent the negative effects of MAC-address randomization can be mitigated to increase the accuracy of device identification, along with other techniques. Others are also developing techniques that may prove to be useful [18]. In addition, it remains unclear to what extent MAC randomization is effective [8,21], and to what extent it actually affects accurately counting pedestrian dynamics. For example, if the used random MAC address remains the same during an epoch, footfall counting may still be possible.

5.3 Stationary Versus Nonstationary Devices

As a final challenge, detections are hindered by the fact that many stationary WiFi-based devices mingle with nonstationary devices. When counting pedestrian (flows), the two need to be separated. For flows, this may be simple if we can assume that stationary devices at one location do not show up at another. For footfall counting, which takes place at a single location, ensuring that only nonstationary devices are counted is important.

A simple solution to this problem is to filter out pseudonyms that have been seen for a long time. Unfortunately, such an approach may easily violate *FDP2*, which boils down to keeping information on a detected device too long. Fortunately, we can use our Bloom-filter approach for distinguishing stationary devices. The basic idea is to still register detected pseudonyms in a Bloom filter, but now to count how often a specific entry is set, leading to a counting Bloom filter. Assuming that over time a stationary

device is more often detected than a nonstationary one, we can set a watermark on all entries of the counting Bloom filter to separate what we see as entries belonging to stationary devices and those that do not. We then extract two new Bloom filters: one of which the counts per entry were less than the chosen threshold, and one with entries that counted equal or more than the threshold. Again, all operations can be done on homomorphically encrypted Bloom filters. Eventually, only the encrypted Bloom filter with detections from nonstationary devices is returned, after having shuffled its entries so that only counting can be done.

6 Conclusions

Automatically measuring the dynamics of pedestrians continues to be a difficult problem. It is somewhat surprising to see the optimism that various groups report on attained accuracies, but above all that so few groups have been paying attention to the protection of privacy. We have come to the conclusion that privacy can be successfully protected using a combination of Bloom filters for storing detected pseudonyms, together with homomorphic encryption techniques for combining filters under encryption. At the same time, more work needs to be done, notably when it comes to settings in which sensors have overlapping ranges, or when many Bloom filters need to be combined.

Regardless of our means to protect privacy, we need to be aware of the difficulty of gathering accurate detections. MAC-address randomization is one hindrance, but there are many more, as we have discussed. We speculate that once privacy protection is indeed considered to be safe, we may be able to open paths to smartphone apps that assist in measuring crowd dynamics, operating completely unobtrusively, in the background. This approach is very similar to the apps used for warning a user that he or she was in close range to a COVID-infected person, which used the privacy-protecting protocol Decentralized Privacy-Preserving Proximity Tracing (DP-3T) [19].

References

1. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: theory and implementation. ACM Comput. Surv. **51**(4), 1–35 (2018)
2. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
3. Chilipirea, C., Dobre, C., Baratchi, M., van Steen, M.: Identifying movements in noisy crowd analytics data. In: 19th International Conference Mobile Data Management (MDM 2018), pp. 161–166. IEEE Computer Society Press, Los Alamitos, CA, June 2018
4. Chilipirea, C., Petre, A., Dobre, C., van Steen, M.: Presumably simple: monitoring crowds using WiFi. In: 17th International Conference on Mobile Data Management, pp. 220–225. IEEE, IEEE Computer Society Press, Los Alamitos, CA, June 2016
5. Chilipirea, C., Baratchi, M., Dobre, C., van Steen, M.: Identifying stops and moves in WiFi tracking data. Sensors (Switserland) **18**(11) (2018)
6. Council of the European union: proposal for a regulation of the european parliament and of the council concerning the respect for private life and the protection of personal data in electronic communications and repealing directive 2002/58/EC (Regulation on Privacy and Electronic Communications). ST 5008 2021 (2021)

7. Draghici, A., van Steen, M.: A survey of techniques for automatically sensing the behavior of a crowd. ACM Comput. Surv. **51**(1), 1–40 (2018)
8. Fenske, E., Brown, D., Martin, J., Mayberry, T., Ryan, P., Rye, E.: Three years later: a study of MAC address randomization in mobile devices and when it succeeds. Proc. Priv. Enhancing Technol. **2021**(3), 164–181 (2021)
9. Ferguson, N., Schneier, B., Kohno, T.: Cryptography Engineering: Design Principles and Practical Applications. John Wiley, New York (2010)
10. Groba, C.: Demonstrations and people-counting based on Wifi probe requests. In: 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), pp. 596–600 (2019)
11. Lai, Y., Kontokosta, C.: Quantifying place: analyzing the drivers of pedestrian activity in dense urban environments. Landscape Urban Plan. **180**, 166–178 (2018)
12. Martella, C., Li, J., Conrado, C., Vermeeren, A.: On current crowd management practices and the need for increased situation awareness, prediction, and intervention. Saf. Sci. **91**, 381–393 (2017)
13. Papapetrou, O., Siberski, W., Nejdl, W.: Cardinality estimation and dynamic length adaptation for bloom filters. Distrib. Parall. Databases **28**(2), 119–156 (2010)
14. Southworth, M.: Designing the walkable city. J. Urban Plan. Dev. **131**(4), 246–257 (2005)
15. Stanciu, V.D., van Steen, M., Dobre, C., Peter, A.: k-anonymous crowd flow analytics. In: MobiQuitous 2020–17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp. 376–385, December 2020
16. Stanciu, V.D., van Steen, M., Dobre, C., Peter, A.: Privacy-preserving crowd-monitoring using bloom filters and homomorphic encryption. In: Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking, pp. 37–42. ACM Press, New York, NY, April 2021
17. Swamidass, S.J., Baldi, P.: Mathematical correction for fingerprint similarity measures to improve chemical retrieval. J. Chem. Inf. Model. **47**(3), 952–964 (2007)
18. Torkamandi, P., Kärkkäinen, L., Ott, J.: An online method for estimating the wireless device count via privacy-preserving Wi-Fi fingerprinting. In: Hohlfeld, O., Lutu, A., Levin, D. (eds.) PAM 2021. LNCS, vol. 12671, pp. 406–423. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72582-2_24
19. Troncosa, C., et al.: Decentralized privacy-preserving proximity tracing. CoRR abs/2005.12273 (2020)
20. Uras, M., Cossu, R., Ferrara, E., Bagdasar, O., Liotta, A., Atzori, L.: Wi-Fi probes sniffing: an artificial intelligence based approach for MAC addresses derandomization. In: 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), pp. 1–6. IEEE (2020)
21. Vasilevski, I., Blazhevski, D., Pachovski, V., Stojmenovska, I.: Five years later: how effective is the MAC randomization in practice? The no-at-all attack. In: Gievska, S., Madjarov, G. (eds.) ICT Innovations 2019. CCIS, vol. 1110, pp. 52–64. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33110-8_5
22. Wang, M., Zhou, J., Long, Y., Chen, F.: Outside the ivory tower: visualizing university students' top transit-trip destinations and popular corridors. Reg. Stud. Reg. Sci. **3**(1), 202–206 (2016)
23. Wijermans, N., Conrado, C., van Steen, M., Li, J., Martella, C.: A landscape of crowd-management support: an integrative approach. Saf. Sci. **86**(7), 142–164 (2016)
24. Zhou, J., Wang, M., Long, Y.: Big data for intra-metropolitan human movement studies: a case study of bus commuters based on smart card data. Int. Rev. Spat. Plan. Sustain. Dev. **5**(3), 100–115 (2017)