Anonymized Counting of Nonstationary Wi-Fi Devices When Monitoring Crowds

Valeriu-Daniel Stanciu
v.stanciu@utwente.nl
University of Twente
Enschede, The Netherlands

Maarten van Steen
m.r.vansteen@utwente.nl
University of Twente
Enschede, The Netherlands

Ciprian Dobre
ciprian.dobre@upb.ro
University Politehnica of Bucharest
Bucharest, Romania

Andreas Peter
andreas.peter@uol.de
University of Oldenburg
Oldenburg, Germany
University of Twente
Enschede, The Netherlands

ABSTRACT

Pedestrian dynamics are nowadays commonly analyzed by leveraging Wi-Fi signals sent by devices that people carry with them and captured by an infrastructure of Wi-Fi scanners. Emitting such signals is not a feature for devices of only passersby, but also for printers, smart TVs, and other devices that exhibit a stationary behavior over time, which eventually end up affecting pedestrian crowd measurements. In this paper we propose a system that accurately counts nonstationary devices sensed by scanners, separately from stationary devices, using no information other than the Wi-Fi signals captured by each scanner in isolation. As counting involves dealing with privacy-sensitive detections of people's devices, the system discards any data in the clear immediately after sensing, later working on encrypted data that it cannot decrypt in the process. The only information made available in the clear is the intended output, i.e. statistical counts of Wi-Fi devices. Our approach relies on an object, which we call *comb*, that maintains, under encryption, a representation of the frequency of occurrence of devices over time. Applying this comb on the detections made by a scanner enables the calculation of the separate counts. We implement the system and feed it with data from a large open-air festival, showing that accurate anonymized counting of nonstationary Wi-Fi devices is possible when dealing with real-world detections.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Security and privacy** → **Privacy protections**.

KEYWORDS

crowd monitoring, statistical counting, anonymized counting, nonstationary devices, privacy protection, Bloom filters, homomorphic encryption

ACM Reference Format:
Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. 2022. Anonymized Counting of Nonstationary Wi-Fi Devices When Monitoring Crowds. In *Proceedings of the International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '22), October 24–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3551659.3559042

MSWiM '22, October 24–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9479-6/22/10.
https://doi.org/10.1145/3551659.3559042

1 INTRODUCTION

With a continuously increasing desire for uninterrupted connectivity, most people nowadays carry with them a smartphone whenever they leave their house. Besides offering people access to the Internet, smartphones leave radio traces behind them wherever they go. For example, Wi-Fi interfaces of smartphones, whenever enabled, periodically broadcast radio signals known as probe requests to discover available nearby Wi-Fi networks. These signals can be easily captured by Wi-Fi scanners placed in public spaces; interpreted and later aggregated into statistical counts, they are being leveraged into a tremendous source of behavioral information regarding the dynamics of the people carrying the emitting devices. Such constructions are called *Wi-Fi based crowd-monitoring systems* (CMS), large-scale deployments being already implemented in many cities across the globe.

Along with signals transmitted by devices of passersby, CMSs receive signals coming from devices not belonging to the crowd intended for monitoring. There are fixed devices such as printers, smart TVs, as well as many other home appliances from neighboring buildings, which are Wi-Fi enabled. Also, there are devices that are not necessarily fixed, yet they are not part of the crowd either, such as laptops or even smartphones of people living or working in nearby buildings, displaying a stationary behavior.

As the focus of a CMS is pedestrian dynamics, stationary devices end up negatively influencing crowd measurements. Strategies for setting them apart usually rely on fingerprinting Wi-Fi devices over time by making use of information extracted from probe requests, e.g., MAC addresses, received signal strength (RSS), frequency of probing, etc. These fingerprints uniquely identify devices belonging to individuals (i.e. natural persons), thus raising serious privacy concerns. More precisely, such strategies are prone to profiling allegations, as profiles of individuals can be potentially created as a side effect, without consent, in the process, a practice frowned upon by privacy watchdogs and strictly regulated by data protection regulations such as the GDPR in the EU.

Modern CMS proposals follow privacy by design principles and perform *anonymization on the fly*. This process happens directly on scanners, and it implies discarding privacy-sensitive data as soon as possible after ingestion, allowing only processed privacy-friendly data that is sufficient to serve the intended purpose of the system, i.e. statistical counts on crowds. In other words, by construction, such a CMS would not be allowed to maintain data for building

fingerprints of devices. Therefore, a novel method is needed to support the separate counting of nonstationary and stationary devices, achieving the same goal as it was achieved through fingerprinting but without needing access to privacy-sensitive data.

We build upon the notion of t-persistence [8] and we call *nonstationary* and *stationary* devices the devices whose probe requests reach a certain scanner in less than t, respectively at least t out of a total of c_e epochs (i.e. predefined fixed-length time intervals) preceding the concerned moment of counting. We propose a system that allows separately counting these two types of devices by operating solely on encrypted data that cannot be decrypted in the process. This is made possible by making use of an object, which we call *comb*, that maintains an encrypted representation of the frequency of occurrence of devices over time. We implement the system and evaluate it using real-world data from a large open-air festival, achieving a mean accuracy of 99.9% when counting nonstationary devices sensed by the most crowded scanner throughout all the epochs in the dataset.

The rest of this paper is structured as follows. Section 2 presents background information and reviews the related work. Section 3 introduces the system model. In Section 4 we propose our construction, followed by an evaluation in Section 5 and a discussion in Section 6. Finally, Section 7 concludes the paper.

2 BACKGROUND & RELATED WORK

Devices with an active Wi-Fi interface periodically broadcast wireless signals (i.e. 802.11 Management Frames) called *probe requests*, expecting to receive back *probe responses* from access points (APs) available in their vicinity, responses that contain information necessary for a potential connection. Probe requests happen outside any established connection, so they circulate in the clear. Moreover, any Wi-Fi scanner can receive them since they are being broadcasted. Therefore, they represent a rich source of information that can be easily sniffed.

The header of a probe request frame contains, among other information, the sender's MAC address, serving as an identifier of the sending device. By having a scanner performing Wi-Fi sniffing at a location over time (e.g., a location where pedestrian traffic is expected), one can get a good idea regarding the devices passing through that scanner's range. Considering that most people nowadays carry Wi-Fi enabled devices, the step from sensing devices to monitoring crowds came naturally, as it was proved that a clear correlation can be seen between measured devices and people present in a certain location [18]. Such measurements, commonly known as *statistical counts*, are the expected outputs of a CMS, and they can lead to accurate representations of crowds under the assumption that an appropriate correction factor is applied [5, 19].

A common crowd-monitoring scenario is presented in Fig. 1. An entity providing crowd-monitoring services (i.e. a service provider) runs an infrastructure of scanners, generally managed by a server. Each scanner gathers Wi-Fi signals and creates a list of devices detected within a certain period of time (i.e. epoch). It passes that list to a server, which is later queried by a party interested in statistical counts on crowds (i.e. a consumer). A classical query is that of footfall, asking for how many devices were detected in the range of a scanner in an epoch. An improved version of the query, which we also aim to accommodate in this paper, can ask for more insights, such as how many of those devices displayed a nonstationary versus a stationary behavior.

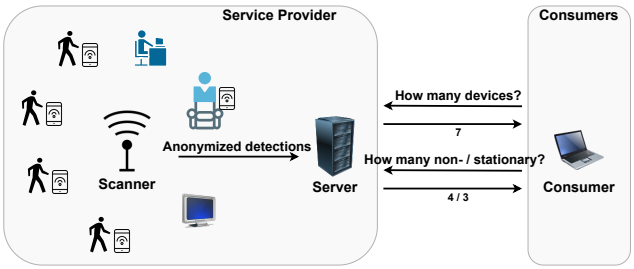

Figure 1: Intended behavior of a CMS offering statistical counts on footfall, including the ability to count separately nonstationary and stationary devices.

Anonymization on the fly is a vital prerequisite for protecting collected data generated by individuals' devices from being exposed to privacy-invasive situations. In CMSs implementing this concept [3, 20, 21], scanners process the data sensed from devices into an anonymized format that still allows the computation of statistical counts. Immediately afterwards, they discard the original data. As a result, only anonymized data leaves the scanners at the end of each epoch. This can be achieved, for example [20], by immediately encoding data into a format facilitating statistical counts and encrypting it with a cryptographic scheme that allows the computation of those counts under encryption while leaving no possibility of decryption through the process. In such a system, once a device is recorded as present within an epoch, it remains counted, so discarding detections on the fly does not affect the statistical counts. However, being able to tell whether a sensed device is nonstationary or stationary can prove to be challenging, as a single spotting of a device may not be enough for making a decision.

There are different methods researchers attempted to use for discriminating between nonstationary and stationary devices based on transmitted Wi-Fi signals. Chilipirea et al. [9] investigated the use of so-called *stay points* [14]. Essentially, the approach relies on the assumption that a nonstationary device will be detected by multiple nonadjacent scanners. We cannot use such an approach since it implies a post-factum decision based on information from multiple scanners, whereas in our case the decision should be made directly on the scanner, in isolation, while having no access to information external to the concerned scanner.

A solution that can indeed be deployed in isolation on a scanner is presented by Redondi et al. in [16]. The authors extract, for each MAC address seen in probe requests, features such as interprobe period, RSS, number of broadcasted or directed probe requests, etc. They use these features to build a machine-learning algorithm for classifying devices into nonstationary (handheld) and stationary (nonhandheld) devices. However, for such a system to be effective, privacy-sensitive data must live for a much longer period than anonymization on the fly permits.

Lastly, single-shot attempts such as [2] try to make the distinction by looking only at the MAC address of a device, more precisely at the first 24 bits representing the Organizationally Unique Identifier

(OUI). The advantage of such an approach is that, besides being applicable in isolation, it does not need to store data for a long time. There are, though, several drawbacks, such as the fact that new OUIs are constantly being assigned, therefore additional effort is required to maintain an up-to-date list, and, more importantly, the same OUI can be very well assigned by manufacturers to devices of both types, thus making the approach impractical for our purpose.

3 SYSTEM MODEL

3.1 Overview

We model a CMS as a system run by a *service provider* (SP), offering crowd-monitoring insights in the form of statistical counts to interested parties, which we call *consumers* (see Fig. 1). The SP manages an infrastructure of *scanners*, which detect Wi-Fi devices in their vicinity. Scanners collect and group such detections in sets corresponding to predefined periods of time called *epochs*. Consumers can address queries to the SP, asking for statistical counts such as footfall found near a scanner in a specific epoch. Furthermore, consumers can ask for more granular information, such as how many of the counted devices are (non)stationary. The system should deliver its functionality in such a way that the privacy of individuals whose devices are detected is not compromised in the process.

3.2 Formalities

We denote by $\mathcal{S} = \{s_1, ..., s_n\}$ the set of all scanners managed by the SP. Each scanner $s \in \mathcal{S}$ performs Wi-Fi sensing across successive time intervals called *epochs*.

Definition 1. An **epoch** $e \in \mathcal{E}$ is a time interval having $t_{start}(e)$ as beginning and $t_{end}(e)$ as end, where $\mathcal{E}$ denotes the set of all such epochs.

When a scanner receives a probe request from a nearby device, it reads the MAC address $a \in \mathcal{A} \subset \{0,1\}^{48}$ encased in the probe request and assigns it, according to the timestamp of reception t_r, to the corresponding epoch e for which $t_{start}(e) \leq t_r < t_{end}(e)$.

Definition 2. We call **detection** a triplet (a, s, e) indicating that a device bearing the MAC address a was detected by scanner s during epoch e. We denote the set of all such detections made by a scanner s during an epoch e as $\mathcal{D}_{s,e}$.[1]

Typically, a CMS is able to offer information regarding footfall in the range of a scanner s during an epoch e by computing the statistical count $|\mathcal{D}_{s,e}|$. However, in this paper we go further and aim to offer additional valuable information for crowd monitoring, such as how many of the spotted devices are *(non)stationary*.

Definition 3. For a current epoch e, a set of c_e consecutive epochs $E = \{e_{-c_e}, ..., e_{-1}\}$ preceding it and a threshold t, we define a **nonstationary** device as a device detected in an epoch e near a scanner s that was also detected by the same scanner s in less than t out of the c_e epochs in E. Conversely, a device is **stationary** if it was detected by s in at least t out of the c_e epochs in E.

[1]Note that by using sets, we avoid counting the same device multiple times within an epoch. This is useful since many Wi-Fi devices are known to transmit numerous probe requests in short periods of time, while for a CMS it is sufficient that a device signals its presence once in an epoch to count it.

For an epoch e and a scanner s we denote the sets of nonstationary and stationary devices as $\mathcal{ND}_{s,e}$ and $\mathcal{SD}_{s,e}$, respectively. The corresponding statistical counts can be, thus, computed as $|\mathcal{ND}_{s,e}|$ and $|\mathcal{SD}_{s,e}|$. These, together with $|\mathcal{D}_{s,e}|$, represent the types of outputs the system should offer.

3.3 Threat model

Throughout the crowd-monitoring process, the system senses and manages data generated by Wi-Fi devices, many of them belonging to *people* from the crowd. Such data is privacy-sensitive and must be carefully handled. We consider an attacker having as main purpose learning privacy-sensitive information about the individuals being sensed. To reach her target, the attacker could compromise each component of the system and, without deviating from the protocol, try to infer as much insights as possible from the data handled by that component. Such an attacker is commonly known as *honest-but-curious* (HBC). To make sure that such an attack cannot succeed, we demand three main security goals to be met, while ensuring that one assumption is followed.

Anonymization on the fly. There should be no data in the clear surviving more than the duration of the epoch in which it was generated, neither outside the scanner that handles it. Gathering and processing detections is, thus, confined to each scanner and limited in time, allowing nothing else than anonymized data to leave the scanners. Note that in order to ensure that this procedure is performed correctly, we need to demand that scanners are tamper-proof.

Blind server. The server should not store, nor handle privacy-sensitive data that it can understand. This requirement offers protection against SPs who could try to infer additional information from the data they handle. Nevertheless, we assume that the server executes its tasks correctly.

Outputs. The system should allow consumers to learn statistical counts on crowds, as this is the intended functionality of the system, but nothing else. Also, the system should enroll only consumers that have a publicly verifiable identity, such as a public key certified by a trusted certificate authority.

Noncolluding entities assumption. The SP does not collude with any of the enrolled consumers, this being a common request of multi-party computation constructions. In essence, SP and consumers are not allowed to cooperate outside the protocol for mutual information enrichment. This also implies that the SP is not allowed to enroll itself as a consumer.

4 OUR CONSTRUCTION

In this section, we start by presenting a state-of-the-art existing method for counting footfall in a privacy-preserving way. On top of it, we introduce a novel mechanism for separating, on the spot, sensed devices into nonstationary and stationary, as demanded by their definition. In this process we make use of an instrument obliviously built under encryption that makes the distinction possible and that we also introduce in this section.

4.1 Statistical counting with Bloom filters

For computing statistical counts on crowds, Bloom filters (BFs) have been proposed [20]. BFs [7] are probabilistic data structures used for representing sets in a space-efficient way. They consist of arrays with m positions, initially all set to 0, defined along with k different hash functions. To add an element in the set, the k hashes of the element are computed and the positions in the BF corresponding to the results are set to 1. Checking whether an element is a member of the set is done similarly, by computing the k hashes and verifying that corresponding BF positions are all set to 1. False negatives are not possible, since if an element was written in the BF, the positions definitely remain set to 1. However, false positives are possible since a position can be set to 1 by the hashes of different elements, a probability of false positives p being expected when n elements are present in the BF.

We leave security requirements aside for a moment to make clear the functionality under the hood. We will get back to them in subsection 4.3, where we present a multi-party cryptographic construction fulfilling these requirements, along with a detailed description of the actions executed by each party.

In the context of crowd monitoring, the set of detections in an epoch is encoded into a BF. Later on, based on the count of 1's in the BF c_t, one can get an estimation c of the cardinality of the original set of detections, as Swamidass and Bald propose in [22], by computing the formula in eq. (1). This estimation is highly accurate, as shown by Papapetrou et al. in [15].

$$c = -\frac{m}{k} \ln\left(1 - \frac{c_t}{m}\right) \tag{1}$$

Thus, for a scanner s and an epoch e, the computation on the corresponding BF of $c \approx |\mathcal{D}_{s,e}|$.

4.2 Combing: Separately counting nonstationary from stationary devices

Estimating statistical counts using eq. (1) is intended for footfall insights. However, the computed values of c cover the sensed devices altogether, including stationary devices that are not part of the actual footfall, whereas footfall is much better represented by the nonstationary devices alone.

An ideal solution would be able to simply tell nonstationary from stationary devices detected in an epoch, as indicated by the choice of t and c_e in Definition 3, and write them in two different BFs. Then, the corresponding granular statistical counts could be computed by separately applying eq. (1) on the two BFs. Yet, in our case, we are dealing with a single BF containing all the detections in an epoch. We aim to start from the bottom up and leverage this single BF into something close to the two BFs in the ideal case above.

The BF-equivalent of a device being detected in an epoch is represented by the k positions indicated by the hashes computed on its address. If the same device is detected by the same scanner across multiple epochs, still the same k positions will correspond to it. Intuition says that positions corresponding to stationary devices will be written in BFs, over time, more often than positions corresponding to nonstationary devices. This leads us to envisioning an object called a *comb* to help us make this separation.

Definition 4. For a scanner s, an epoch e and a BF of length m containing the detections made by s during e, we define the **comb** as an array of the same length m, for which each position indicates whether the corresponding position in the BF should be taken into account when counting nonstationary or stationary devices. The comb is built across c_e epochs preceding e by summing up, positionwise, the BFs built at s during those epochs; the result is similar to a *counting BF* [12].

We present in Fig. 2 an example of a combing process when considering nonstationary devices as those being detected by a scanner s in less than 20 out of the 24 epochs preceding the current epoch e. Applying the comb on a BF produces two BF-like structures corresponding to supposedly nonstationary and stationary devices. Subjecting these structures to eq. (1) generates the estimated counts nc and sc, which approximate the statistical counts $|\mathcal{ND}_{s,e}|$ and $|\mathcal{SD}_{s,e}|$.

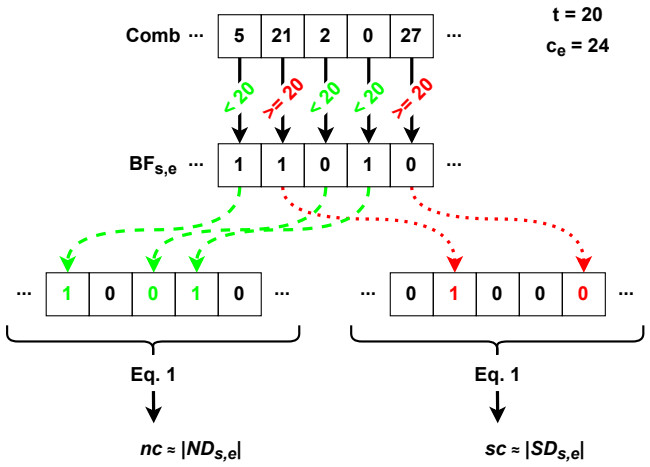

Figure 2: Combing a BF for $t = 20$, $c_e = 24$, in order to compute the statistical counts of nonstationary and stationary devices seen by scanner s during epoch e.

We stress that these structures are not BFs by definition, as they are generated according to some conditions that do not necessarily translate into a separation of elements but rather into a *separation of positions*. Their statistical properties (e.g., the number of 1's), though, are relevant for statistical counts. Yet, the accuracy of the estimations may be different from in the ideal case (i.e. separate actual BFs), as we know that more elements can be hashed to the same position and deciding how to label that position (i.e. nonstationary or stationary) can potentially have an impact. We will analyze these aspects in detail in Section 5.

4.3 Anonymized counting under encryption

BFs, despite being different from detections in the clear, their representation of data still leaves the stored 0's and 1's visible. As the identifier space (i.e. MAC address space) is easily enumerable [6], BFs storing such elements are susceptible to brute-force attacks, in which an attacker can check, in limited time, the presence of each possible identifier by iteratively computing the k hash functions

and verifying the corresponding positions. Therefore, BFs should not be allowed to live as they are for more than an epoch, nor outside the scanner that generates them. Still, in order to build a comb, we need to combine data from multiple epochs, data that should have been already discarded. To overcome this problem, we consider the option of encrypting data before discarding it in such a way that it allows the operations that we need to perform on it, but this time under encryption.

Homomorphic encryption [17] is a type of encryption that allows mathematical operations directly on the encrypted data, without the need for decryption. The results under encryption are the same as if the operations were performed on data in the clear. In particular, in our system, in order to build the comb, we need an encryption scheme allowing additions under encryption. Also, to be effective for BFs, which only contain 0's and 1's, the scheme should be probabilistic, such that encrypting the same value multiple times yields different ciphertexts. Lastly, the scheme should be asymmetric, such that an SP in possession of a public key can encrypt and perform operations under encryption to address consumer queries, but that only the intended consumer holding the corresponding secret key can decrypt the received results. ElGamal [11] is a scheme bearing all these properties, and we choose to use the additively homomorphic version of it in our system.

Let us now assemble the components and present how the whole process of anonymized counting takes place.

Preamble. Consumers enroll in the system by presenting their public key to the SP, which stores it on the server and forwards it to the scanners.

Sensing. Each epoch, scanners perform sensing and write detections in a BF. At the end of an epoch, they encrypt a copy of the BF, positionwise, for each enrolled consumer, using their public key. They discard the original BF and send the resulting encrypted BFs (EBFs) to the server.

Querying. A consumer interested to find out insights on footfall in the vicinity of a scanner s within an epoch e informs the SP of her interest. She specifies, along with the query, the number of epochs c_e preceding epoch e for which she would like to have a comb built.

Response. The response to a query is prepared by the SP on the server. The server generates the comb by summing up positionwise, under encryption, the EBFs generated by s in the c_e epochs preceding e, making use of the homomorphic property of the encryption scheme. It delivers, as a response, the EBF from scanner s and epoch e, along with the generated comb. Before that, it shuffles the positions of both structures to make sure that any BF-related meaning is lost. Note, though, that the shuffling of the comb should mirror the shuffling of the EBF, because the order of the positions may not be important for combing, but the correspondence between the positions of the two is still needed.

Result. The consumer, being in the possession of the secret key, decrypts the shuffled EBF and comb, performs the combing according to a threshold t that she desires and applies eq. (1) to estimate the statistical counts.

5 EVALUATION

In this section, we start by running a set of preliminary experiments, testing our intuition that positions in a comb corresponding to stationary devices are set to 1 more often than positions corresponding to nonstationary devices, thus allowing their separation based on a threshold. We continue by presenting an error analysis to understand how to setup the system in order to minimize counting errors. Then, we perform an evaluation using a real-world dataset to see how well the system can separately count nonstationary from stationary devices. Finally, we do an actual implementation, including the encryption layer, and analyze its performance.

5.1 Preliminary experiments

BFs are generally configured to support a number of inserted elements n while satisfying a desired probability of false positives p. To meet these conditions, the length m of the BF should be calculated as $-n \ln p/(\ln 2)^2$ and the optimal number of hash functions k as $-\log_2 p$. For example, to accommodate a maximum of 100 detections at a probability of false positives of 0.01, m should be 959 and k should be 7. In this subsection, we stick to these parameters to run some preliminary experiments. For hashing, we use, with different seeds, MurmurHash3 [4], a fast hash function, noncryptographic, but suitable though for our purpose since BFs and combs are going to be encrypted anyway. As identifiers, we generate random MAC addresses coming from a uniform distribution.

The idea of a comb comes from the intuition that positions where stationary devices are mapped in BFs, are written more often than those where nonstationary devices are, making their separation possible based on a threshold. Following this intuition, we run a set of preliminary experiments. We build a comb for $c_e = 24$ epochs, and we choose, for separation, a threshold t of 20 epochs. For this set of experiments, the epoch length is not important, but we choose c_e as 24 having in mind epochs of 5 minutes and, thus, a total interval of two hours for building the comb. We fix the number of devices per epoch, denoted as d, first to 50, then to 100, from which 10 are stationary and the rest nonstationary. We make the stationary devices appear in a random number of epochs between t and c_e. Each epoch we fill, up to d, with nonstationary devices. For $d = 50$, each nonstationary device appears once; for $d = 100$, we make 10 of them appear randomly between 1 and $t - 1$ times and the rest once. We show the results in Fig. 3, where we display the values found in the comb on the x axis and their mean frequency out of 100 runs on the y axis (i.e. how many positions are in the comb for each value).

We can see from both experiments that indeed writings in the comb are concentrated separately, according to the inserted nonstationary and stationary devices. If we were to apply this comb on a BF containing the detections from an epoch, the positions whose corresponding comb values are found to the left of the dashed line would be considered as belonging to nonstationary devices, whereas those on the right would be considered as belonging to stationary devices.

Yet, we can see a difference between the results of the two experiments. For $d = 50$, the threshold clearly separates the values corresponding to the two types of devices, as the frequency is 0 for values such as 17, 18 and 19. For $d = 100$, although apparently

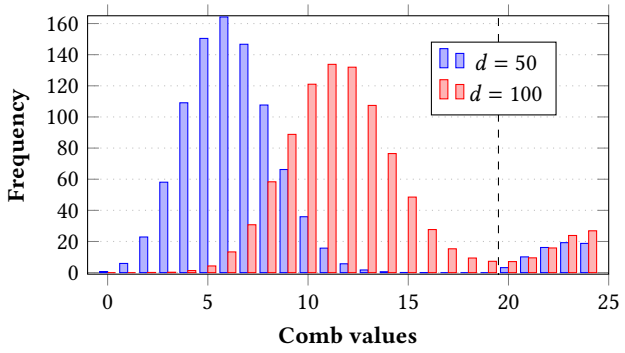

Figure 3: Frequency of comb values when sensing 50 and 100 devices per epoch for $c_e = 24$, using BFs configured for $n = 100$ and $p = 0.01$. The dashed line marks the threshold $t = 20$.

this was the figure for which the system was configured, we can see the curve corresponding to nonstationary devices expanding, through some of its positions, beyond the dashed line, into the territory where positions are marked as stationary. This overlap can incur errors on the counts, and it can visibly get even bigger for lower thresholds or when there are more nonstationary devices appearing more often. Opting for a value of m higher than is usually computed would alleviate the problem, and it comes natural since the comb combines detections from multiple BFs, leading to much more *collisions* than expected for a single BF. We elaborate on this matter below.

5.2 Error analysis

We stated earlier that a higher value for m should be used. In principle, with an infinite m, each device, no matter its type, will write at positions never written by any other devices. As a result, the values at the positions in the comb would be *exactly equal* to the number of occurrences of the devices pointing to them. Thus, applying the threshold would make a perfect separation, as if we would have used from the start two separate BFs for nonstationary and stationary devices. In practice though, we cannot choose m infinite for performance reasons that we will further explain in Section 5.4, so we expect to see comb positions written by different devices. We call such an event a *collision*.

We remind that statistical counts of nonstationary and stationary devices are computed by applying eq. (1) on the two BF-like structures resulting after combing. The only thing from the formula that makes the counts differ from those in the ideal scenario (i.e. two separate BFs) is the change of c_t (i.e. c_{ts} for stationary and c_{tn} for nonstationary) inflicted by values of 1 being misplaced by the comb into the other BF-like structure due to collisions. We present how combing happens at position level in Fig. 4.

Depending on what devices generate them, we have the following taxonomy of elementary collisions: (1) between stationary devices, (2) between a stationary and a nonstationary device and (3) between nonstationary devices. Each of the three can affect in different ways the counts of nonstationary and stationary devices, also linked with which device writes (or not) at a collision-related position in the BF to which the comb is applied.

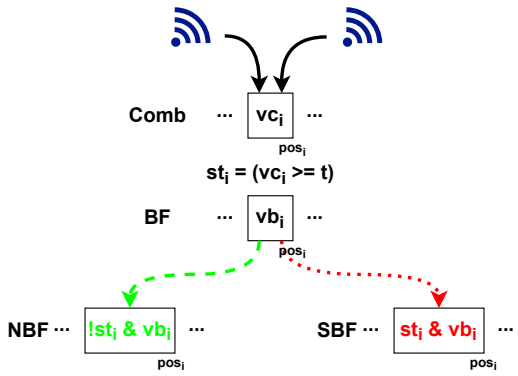

Figure 4: Moving value vb_i from a colliding position i in a BF to the nonstationary (NBF) or stationary (SBF) BF-like structure, based on the relationship between its corresponding value in the comb vc_i and t.

(1). When two stationary devices generate the collision on the comb, the concerned position will be marked as stationary as both devices appeared at least t times and their combined writings will definitely lead to a sum at least as large as t. No matter which of the two devices appears (or not) in the analyzed epoch, combing will not produce any change in c_{ts}, nor in c_{tn}, and, thus, no impact on sc, nor on nc, as long as there is no other nonstationary device writing at that position. In case any nonstationary device writes at that position, c_{tn} will decrease by 1; in addition, in this particular case, if none of the stationary devices appears, c_{ts} will increase by 1.

(2). A position where a stationary and a nonstationary device collide will always be marked as stationary. As in the above case, the presence of another device writing at the same position with a stationary device can only increase the already greater or equal with t sum. If there is no nonstationary device writing at that position in the analyzed epoch, c_{ts} and c_{tn} will not change, no matter whether the stationary device appears or not. If any nonstationary device writes at that position, c_{tn} will decrease by 1 as the position is marked as stationary; moreover, if in this situation the stationary device does not show up, c_{ts} will increase by 1, falsely believing that the device was present.

(3). When two nonstationary devices collide on a position, the position can be marked as nonstationary if the count of epochs in which at least one of them appears is lower than t, otherwise, the position is marked as stationary. The lower the t, the higher the chance of marking the position as stationary in case of such a collision. If the position is marked as nonstationary, there will be no impact whatsoever, no matter what nonstationary devices write at it. Note that stationary devices cannot be expected to write at that position, otherwise they should have been part of the collision. If the position is marked stationary and at least one nonstationary device writes at that position in the analyzed epoch, c_{tn} will decrease by 1 and c_{ts} will increase by 1.

To summarize, when collisions occur, c_{ts} tends to increase and c_{tn} tends to decrease. The systematic effect of this is a potential

overcounting of stationary devices and undercounting of nonstationary devices.

We see that the choice of t, which is a functional parameter dictated by the consumer and her functionality needs, can, depending on t, influence the accuracy of counts in case of collisions between nonstationary devices. We will have this in mind when evaluating the accuracy of the system. However, most of the impact on counts can be prevented by minimizing the probability of having collisions in the first place. This can be done through a careful choice of BF parameters.

We have already mentioned that a higher m is desirable, deviating from typical BF configurations, which choose m and k to match a probability of false positives p. Though, the probability p' that a position in the comb corresponds to a collision is different from p and is the same as the probability that at least two elements write at that position, which we display in eq. (2).

$$p' = \left(1 - \left(1 - \frac{1}{m}\right)^k\right)^2 \tag{2}$$

As our intention is to minimize p' and not necessarily to match p, besides already fixing m as high as performance requirements allow, k should be always chosen as 1.

5.3 Evaluation with real-world data

We proceed with an evaluation using real-world data, to assess how well our mechanism is capable of separately counting nonstationary from stationary devices when faced with real detections sensed by an actual infrastructure of scanners. We are using a dataset collected in 2017 by 30 scanners placed in the city of Assen, The Netherlands. Scanning took place for 12 consecutive days, covering the whole period of a large open-air festival. Scanners were placed on the streets of the city center, gathering detections of devices belonging to people in the outdoor crowds, as well as devices in nearby buildings, capturing, thus, a wide range of stationary and nonstationary behaviors. There were 26 million detections of devices bearing 176 thousand different identifiers (i.e. MAC addresses run through a one-way cryptographic hash function).

We fix the epoch length to 5 minutes, as it proved to be long enough to ensure capturing probe requests from most devices simultaneously present near a scanner [13]. For this epoch length, with few exceptions, detection sets from the dataset consist of less than 1000 devices. Normally, when setting up BFs, for $n = 1000$ and a low p, e.g., 0.01, m should be ≈ 10000 and $k = 7$. Nevertheless, we increase m to 100000 (i.e. 10 times higher) and use $k = 1$, as discussed in Section 5.2. We will later show in Section 5.4 that this high value of m still allows even resource-constrained scanners to produce EBFs for at least two consumers within 5 minutes. Lastly, we fix c_e to 24 (i.e. 2 hours). Two hours of detections should provide enough information to decide whether a device is nonstationary or stationary, in the context of an open-air urban festival and considering pedestrian dynamics in such conditions.

For the following experiments, we consider a scanner placed in the most crowded area of the festival, which gathered a total of 1.6 million detections. For each epoch, we group these detections in detection sets that we encode into BFs. Then, for each BF, we create its associated comb corresponding to the previous c_e epochs. In parallel, we calculate and store the actual frequency of occurrence for each device that we write in the comb. To evaluate the accuracy of counts of nonstationary and stationary devices from an epoch, we compare two things: (1) the results, rounded to the nearest integer, obtained by combing that epoch's BF (i.e. using our mechanism) with (2) the counts obtained by separating devices sensed in that epoch based on the previously stored actual frequency of occurrences (i.e. obeying Definition 3). Note that there may be cases when carry-on devices end up being considered stationary when, for example, they spend more than t epochs in one place. This is consistent with our evaluation because, by definition, those devices are indeed stationary.

We select a sequence of five of the very crowded encountered epochs, as an extreme scenario seen by the system. Each epoch contains around 1000 detections. We first plot, in Fig. 5, the split between nonstationary and stationary devices counted by using our mechanism, when setting the threshold t to 20, 10 and 5, respectively.

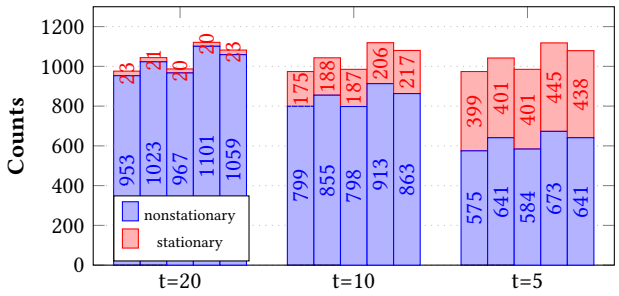

Figure 5: Separate counts of nonstationary and stationary devices for $c_e = 24$ and different values of t.

This figure allows us to understand what the threshold t means for the classification of devices. Choosing a lower t determines the mechanism to consider more devices as stationary, since fewer detections are sufficient for passing the threshold. For $t = 5$ for example, the definition of stationary devices is broad and includes from smart TVs that are present in most epochs to devices that spend 25 minutes in the area and then leave. On the other hand, for $t = 20$ the definition is stricter and, thus, fewer devices are considered stationary. Nevertheless, in our construction the choice of t and the interpretation of what nonstationary and stationary devices are, fall onto the consumer, who proceeds according to her needs; we will come back to this discussion later, in Section 6.1.

For the same sequence of epochs, we want to see how accurate the split we have just presented is. We plot thus in Fig. 6, side by side, actual counts and counts estimated by our system. The lower part of the figure (left y axis) shows nonstationary, while the upper part (right y axis) the shows stationary devices.

For these very crowded scenarios in the dataset and using appropriately chosen system parameters, both nonstationary and stationary devices are estimated with high accuracy. For $t = 20$, the estimated count of nonstationary devices is at most 1 device away from the actual count (i.e. 967 instead of 966 devices leading to an error of 0.1%, equivalent with an accuracy of 99.9%) and for

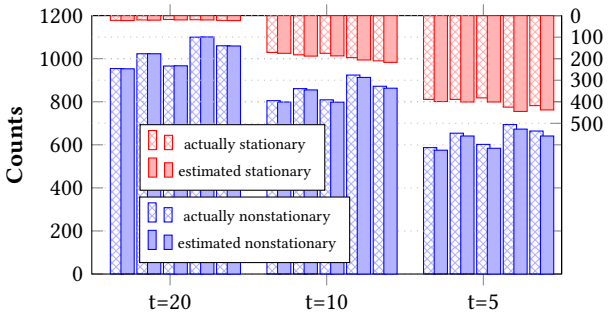

Figure 6: Comparing real counts of nonstationary and stationary devices with counts estimated by the system.

stationary devices at most 2 devices away. For lower t, the system can perform accurate estimations too, though we can notice a small decrease in accuracy. The highest error when estimating nonstationary devices for $t = 5$ is 3.4%, corresponding to an accuracy of 96.6%. This is something we expected to see, as we have shown that a lower t increases the probability of collisions between nonstationary devices, leading to more positions being marked as stationary. To get a better feeling of this, we plot for the same sequence of epochs, in Fig. 7, the absolute error of nonstationary and stationary counts when t equals 20, 10 and 5.

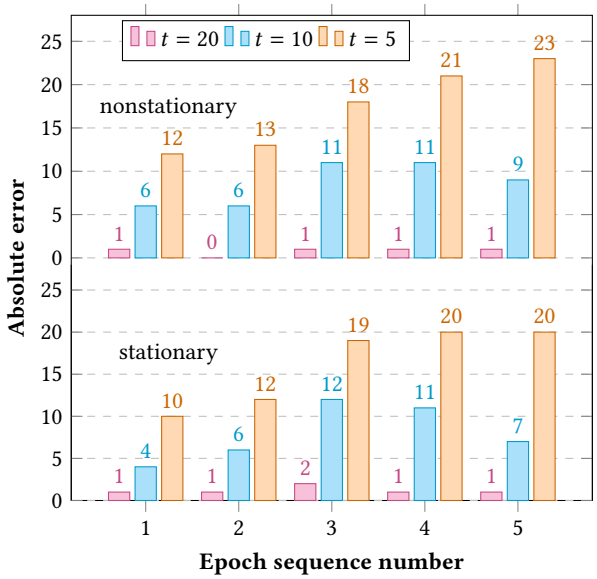

Figure 7: Absolute error of counts of (non)stationary devices.

Constantly, the error is bigger for lower t. We can also see a direct correlation between the decrease of nonstationary counts and the increase of stationary counts. Let us take a look, for example, at the 5th epoch. We interpret the numbers as follows. The collisions in the comb led to a number of positions in the 5th epoch's BF being marked as stationary, despite nonstationary devices wrote there in the comb as well. Not counting these positions as nonstationary diminishes the count of nonstationary devices by 23. Counting the exact same positions as stationary increases the count of stationary devices by 20.

We move on now to analyzing the accuracy of counts of nonstationary devices for the whole period of the festival in the vicinity of the most crowded scanner. Our purpose is to exclude from counts stationary devices that are present most of the time (i.e. printers, smart TVs, home appliances from nearby buildings, etc.). Even though in principle such devices do not move, there may be situations when they do not transmit probe requests for a while or the probe requests do not reach the scanner (e.g., a temporary power cut, a device overload / malfunction, or simply a sudden signal interference). This is why we consider 20 to be an appropriate value for t and we fix it like this. The rest of the parameters of the system remain unchanged, including $c_e = 24$. We present, in Fig. 8, the counts of nonstationary devices estimated by our system throughout all the epochs of the festival, as well as the absolute error for each epoch.

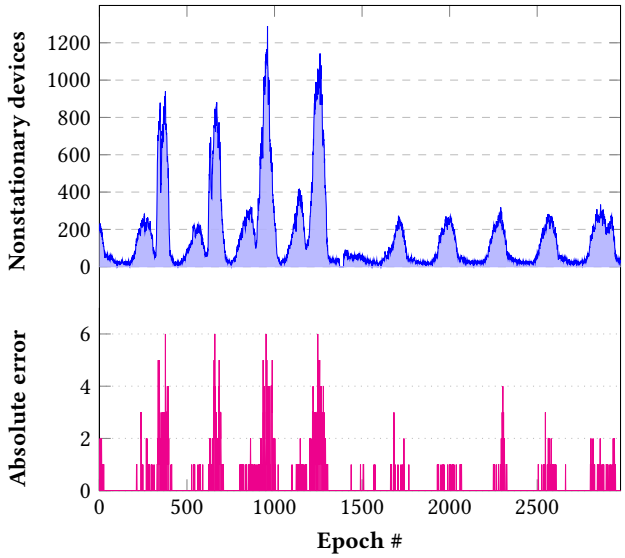

Figure 8: Estimated nonstationary devices and their absolute errors during festival days and afterwards.

The graph covers a total of 2977 epochs spread across 11 days; it does not include the first 24 epochs, as combs need 24 epochs to be built. All estimations are at most 6 devices away from the actual counts. For those epochs with an absolute error of 6 (i.e. 4 of them, very crowded epochs with actual counts between 843 and 1112), accuracy is higher than 99.2% for all of them. The mean accuracy across all the epochs is 99.9%. Overall, in 93.9% of the epochs, the estimation is at most one device away from the actual count; when it is farther than one device away, it is for very crowded epochs where the impact on accuracy is very low. We ran the same experiments for estimating stationary devices and we confirm that the results are similar, i.e. a mean accuracy of 99.6% (apparently lower, but due to stationary devices being fewer and absolute error having thus a higher impact) and estimations at most 5 devices away from actual counts.

5.4 Implementation & Performance analysis

We perform an implementation of the system and assess its performance, covering the procedures done by the different involved parties, including shuffling and operations done under encryption according to the protocol. By going through this complete implementation process, we also validate that implementing such a system is feasible.

Hashing that has to be done to find positions in a BF corresponding to an element can be considered negligible, being in the range of nanoseconds. The same holds for shuffling. The procedures we expect to be the most resource consuming are those using homomorphic encryption. For evaluation, we instantiate ElGamal using the NIST P-256 elliptic curve [1] and use the SCAPI[2] library [10] for homomorphic encryption support.

Scanner. For each enrolled consumer, a scanner must create an EBF at the end of each epoch. Creating an EBF incurs a fixed amount of work, equivalent with m homomorphic encryptions. We implement this on a Raspberry Pi 4B, having a 1.5GHz 64-bit quad-core ARM v8 Cortex-A72 processor, 8 GB of DDR4 RAM memory, a 16 GB microSD memory card and running Ubuntu 20.10 as OS. We perform the encryptions in parallel using C++11 threads. For $m = 100000$, as we used in the evaluation with real-world data, the scanner creates an EBF in 125 seconds. To avoid lagging behind, a scanner should create EBFs for all enrolled consumers faster than the length of an epoch. Thus, for an epoch of 5 minutes, even a resource-constrained scanner such as Raspberry Pi could support 2 enrolled consumers at the same time while providing accurate statistical counts for crowds up to 1000 devices per epoch.

Server. When a consumer launches a query, besides having to deliver the concerned shuffled EBF, the server must create its corresponding comb by performing $(c_e - 1) * m$ additions under encryption. We implement and run this on a basic cloud server with 16GB RAM and a 16-core Intel(R) Xeon(R) Silver 4110 CPU @ 2.10 GHz, running Ubuntu 18.04 x86_64. The additions under encryption are done in parallel using C++11 threads. For $m = 100000$, m additions under encryption are performed in 8.6 seconds; for $c_e = 24$, this means that a comb can be created and delivered to a consumer in approximately 200 seconds. Knowing that the server will have to store EBFs, we check their size. In our implementation, we compute the size occupied by an encryption to 678 B, meaning that, for $m = 100000$, an EBF would occupy 67.8 MB.

Consumer. To be able to compute the statistical counts, the consumer must first decrypt the shuffled EBF and its corresponding comb by performing $2 * m$ decryptions. We consider a consumer having a laptop running Ubuntu 20.04 x86_64, with 8GB RAM and a 4-core Intel(R) Core(TM) i5-10210 CPU @ 1.60 GHz. Such laptop can do the necessary decryptions (also in parallel), for $m = 100000$, in 80 seconds.

6 DISCUSSION

6.1 On choosing c_e and t

In order to separately count nonstationary and stationary devices from an epoch, a consumer must decide on values for c_e and t.

[2]https://github.com/cryptobiu/libscapi

When choosing c_e, a consumer must think about how much time is needed in order to accumulate sufficient information to allow deciding upon the (non)stationary nature of the sensed devices. The choice of c_e (i.e. lower or higher) directly limits the range for t, allowing thus a lower or higher granularity for the interpretation of stationarity. For monitoring crowds, c_e should be chosen to cover more time than people from the crowd typically spend in the sensed area, such that an effective t can be set. For example, if the scanner is placed in a transit area, a lower c_e may prove to be enough. If the scanner is in a place where people tend to spend more time, a higher c_e is preferable.

After choosing c_e, a consumer must set t in order to perform the combing. Before we go into details, it is worth noting again that it can happen that probe requests do not reach the scanner in some epochs due to various reasons, so this should be kept in mind before considering t. The consumer chooses t according to her own interpretation of what a nonstationary or stationary device is. For example, for a t close to c_e, stationary devices will be those that are detected almost always as present and nonstationary all the others. For $t = 3$ on the other hand, nonstationary devices will be mostly those that pass through the scanner's range without spending more than a couple of minutes there and stationary devices all the rest.

A consumer can potentially compute even more sophisticated counts, such as the number of devices spending some time in the range of a scanner but not being almost always present (e.g., by subtracting stationary devices estimated for a t close to c_e from those estimated for $t = 3$). Though, the relevance, as well as the accuracy of such arithmetically estimated counts remain yet to be investigated.

6.2 Security analysis

Our system achieves the security goals proposed in the threat model, under the assumption of noncolluding entities. By achieving these goals, our construction becomes secure against *honest-but-curious* (HBC) adversaries, as we present below.

Detections are encoded into BFs and then immediately discarded at scanner. BFs are encrypted at the end of an epoch, and only then they can leave the scanner. Anonymization on the fly is, thus, satisfied. Moreover, scanners cannot learn anything more than they already see by sensing, as they do not handle any external data.

By handling only encrypted data that it cannot decrypt and by creating combs obliviously under homomorphic encryption, the server is blinded and, thus, cannot learn anything from data dealt with in the process.

Consumers can see, in the clear, shuffled BFs and combs. Such data is meaningless when it comes to the privacy-sensitive detections that were previously stored in it, since it was shuffled and it lost any such meaning in the process. The only meaning left is given by its statistical properties, that is a targeted need for estimating statistical counts.

Security of the system can be broken if the implementation deviates from the system model, the attacker does not follow the protocol (i.e. becoming malicious instead of HBC) or the noncolluding entities assumption is broken. For example, if the server becomes malicious and does not shuffle an EBF, a consumer could find out, through brute-force, the elements encoded in the BF. Also,

if the SP colludes with a consumer, even without being malicious, an HBC server could see what is stored in the EBFs of that consumer.

Finally, our system discards the privacy-sensitive detections and produces only statistical counts on crowds. In future work we plan to investigate whether these counts themselves may allow a possible inference of privacy-sensitive information (e.g., when the monitored crowd consists of a single person) and, if so, what we can do to prevent it (e.g., not producing any result in such cases).

6.3 Limitations on number of consumers

Homomorphic encryption is known to be resource-demanding. With our resources, we were able to set up the system such that it computes highly accurate statistical counts for the considered dataset and for high, as well as low thresholds. However, our limited resources could support a limited number of consumers. For an epoch of 5 minutes, scanners could support 2 consumers, while the server could produce counts for one consumer within an epoch.

More consumers can be supported by the same hardware if m is reduced. We tried to fix m to 10000, therefore supporting 10 times more consumers. Accuracy was still very high for $t = 20$ (i.e. $> 98\%$ for counting nonstationary devices from the sequence of 5 epochs). However, accuracy decreased quicker for lower t's, which we expected since the lower value of m also meant more collisions. To ensure the accuracies obtained by using $m = 100000$ but while supporting more consumers, more powerful hardware is needed.

7 CONCLUSION

In this paper, we proposed a system that can separately count nonstationary from stationary Wi-Fi devices when monitoring crowds. Unlike previous attempts, our system performs the separate counts in an anonymized way, protecting, thus, the privacy-sensitive detections of devices belonging to individuals. The system encrypts and then immediately discards in-the-clear detections, afterwards operating only on encrypted data that it cannot decrypt. As a result, it supports decisions on the stationarity of devices built upon information spanning extended periods of time, which were previously not possible without privacy infringement risks. Moreover, the system allows users to define themselves (and count accordingly) what nonstationary and stationary devices are, based on the frequency of detections in a given period of time and a custom threshold to use for separation.

We implemented the system using a Raspberry Pi as a scanner, a cloud environment as a server and a laptop as a consumer, and we fed it with real-world data from an open-air festival. Our system achieved a mean accuracy of 99.9% when estimating nonstationary devices sensed by a scanner placed in the most crowded area of the festival throughout the whole period of sensing. For the same scanner, 93.9% of the estimations were at most one device away from the actual count. These results show that highly accurate anonymized counting of nonstationary Wi-Fi devices is possible when dealing with real-world detections of crowds and while fully protecting the privacy-sensitive data of the individuals being monitored.

REFERENCES

[1] Mehmet Adalier and Antara Teknik. 2015. Efficient and secure elliptic curve cryptography implementation of Curve P-256. In *Workshop on Elliptic Curve Cryptography Standards*, Vol. 66.

[2] Mikhail Afanasyev, Tsuwei Chen, Geoffrey M Voelker, and Alex C Snoeren. 2010. Usage patterns in an urban WiFi network. *IEEE/ACM Transactions on Networking* 18, 5 (2010), 1359–1372.

[3] Mohammad Alaggan, Mathieu Cunche, and Sébastien Gambs. 2018. Privacy-preserving wi-fi analytics. *Proceedings on Privacy Enhancing Technologies* 2018, 2 (2018), 4–26.

[4] Austin Appleby. 2016. MurmurHash3. (2016). https://github.com/aappleby/smhasher/wiki/MurmurHash3

[5] Lu Bai, Neil Ireson, Suvodeep Mazumdar, and Fabio Ciravegna. 2017. Lessons learned using wi-fi and Bluetooth as means to monitor public service usage. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. 432–440.

[6] Giuseppe Bianchi, Lorenzo Bracciale, and Pierpaolo Loreti. 2012. " Better Than Nothing" Privacy with Bloom Filters: To What Extent?. In *International Conference on Privacy in Statistical Databases*. Springer, 348–363.

[7] Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.

[8] Zhen Cao, Yu-E Sun, He Huang, Hansong Guo, Yang Du, An Liu, and Le Lu. 2020. Finding Persistent Elements of Anomalous Flows in Distributed Monitoring Systems. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1–7.

[9] Cristian Chilipirea, Ciprian Dobre, Mitra Baratchi, and Maarten van Steen. 2018. Identifying movements in noisy crowd analytics data. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 161–166.

[10] Yael Ejgenberg, Moriya Farbstein, Meital Levy, and Yehuda Lindell. 2012. SCAPI: The Secure Computation Application Programming Interface. *IACR Cryptol. ePrint Arch.* 2012 (2012), 629.

[11] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.

[12] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. 2000. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM transactions on networking* 8, 3 (2000), 281–293.

[13] Julien Freudiger. 2015. How talkative is your mobile device? An experimental study of Wi-Fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 1–6.

[14] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. 2005. Extracting places from traces of locations. *ACM SIGMOBILE Mobile Computing and Communications Review* 9, 3 (2005), 58–68.

[15] Odysseas Papapetrou, Wolf Siberski, and Wolfgang Nejdl. 2010. Cardinality estimation and dynamic length adaptation for bloom filters. *Distributed and Parallel Databases* 28, 2 (2010), 119–156.

[16] Alessandro Enrico Cesare Redondi, Davide Sanvito, and Matteo Cesana. 2016. Passive classification of Wi-Fi enabled devices. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 51–58.

[17] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. 1978. On data banks and privacy homomorphisms. *Foundations of secure computation* 4, 11 (1978), 169–180.

[18] Lorenz Schauer, Martin Werner, and Philipp Marcus. 2014. Estimating crowd densities and pedestrian flows using wi-fi and bluetooth. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 171–177.

[19] Balamurugan Soundararaj, James Cheshire, and Paul Longley. 2020. Estimating real-time high-street footfall from Wi-Fi probe requests. *International Journal of Geographical Information Science* 34, 2 (2020), 325–343.

[20] Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. 2021. Privacy-Preserving Crowd-Monitoring Using Bloom Filters and Homomorphic Encryption. In *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*. 37–42.

[21] Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. 2020. k-Anonymous Crowd Flow Analytics. In *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 376–385.

[22] S Joshua Swamidass and Pierre Baldi. 2007. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of chemical information and modeling* 47, 3 (2007), 952–964.