Privacy-Friendly Wi-Fi-Based Crowd Monitoring for Pedestrian Dynamics Analytics

Valeriu-Daniel Stanciu

Privacy-Friendly Wi-Fi-Based Crowd Monitoring for Pedestrian Dynamics Analytics

Valeriu-Daniel Stanciu

PRIVACY-FRIENDLY WI-FI-BASED CROWD MONITORING FOR PEDESTRIAN DYNAMICS ANALYTICS

D I S S E R T A T I O N

to obtain a joint degree, namely
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. ir. A. Veldkamp,
and at the University Politehnica of Bucharest,
on the authority of the rector,
M.C. Costoiu,
on account of the decision of the Doctorate Board
to be publicly defended
on Friday 9 December 2022 at 12.45

by

Valeriu-Daniel Stanciu
born on the 16th of June 1990
in Campina, Romania

This dissertation has been approved by:

Supervisors:

prof. dr. ir. M.R. van Steen	University of Twente
prof. dr. A. Peter	University of Oldenburg
prof. dr. ing. C. Dobre	University Politehnica of Bucharest

DSI Ph.D Thesis Series No. 22-008

ISBN: 978-90-365-5491-6
ISSN: 2589-7721
DOI: 10.3990/1.9789036554916
https://doi.org/10.3990/1.9789036554916

Typeset with LaTeX.
Printed by: Gildeprint
Cover design: Douwe Oppewal

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Graduation Committee:

Chairman/Secretary:	prof. dr. J.N. Kok	University of Twente
Supervisors:	prof. dr. ir. M.R. van Steen	University of Twente
	prof. dr. A. Peter	University of Oldenburg
	prof. dr. ing. C. Dobre	University Politehnica of Bucharest
Committee Members:	prof. dr. ir. G.J. Heijenk	University of Twente
	dr. A. Sperotto	University of Twente
	prof. dr.-ing. D. Reinhardt	University of Gottingen
	prof. dr. S. Klous	University of Amsterdam
	dr. ing. R.I. Ciobanu	University Politehnica of Bucharest

Abstract

Understanding pedestrian dynamics in crowded public spaces has shown to be important. Nowadays, there are widely deployed sensing infrastructures that detect Wi-Fi signals emitted by smartphones carried by people in crowds. Based on these detections, crowd-monitoring insights can be derived in the form of *statistical counts*, offering information such as the footfall in a location as well as crowd flows between several locations. Because detections of devices carried by individuals must be handled in the process, there are legitimate concerns regarding the privacy of those sensed individuals. There have been attempts to address these privacy concerns, but they proved to be insufficient, mostly because uniquely tracing back to individuals still remained possible.

We propose two new methods that protect the privacy-sensitive detections of individuals while still allowing the computation of statistical counts on crowds. The first method anonymizes detections on the fly, ensuring protection under what we call *detection k-anonymity* for all the collected data, no matter how the anonymized data is combined to address future queries. The second method relies on encoding detections into probabilistic data structures called *Bloom filters* (BFs), and then encrypting the resulting BFs with a *homomorphic encryption* (HE) scheme. As part of a multi-party cryptographic construction, HE allows performing the operations needed for computing the statistical counts directly on the encrypted data, without the ability to decrypt, revealing only the end result in the clear to the intended recipient. Furthermore, to enable granular decisions upon which detected devices are considered as part of the crowd and under the same privacy protection guarantees ensured by the combination of BFs with HE, we explore the possibility of separately counting nonstationary from stationary devices based on their frequency of detection.

We implement and extensively evaluate the proposed contributions using simulated, as well as real-world data. Our results demonstrate that highly accurate statistical counting for pedestrian dynamics is possible while privacy protection is guaranteed.

Abstract (Dutch)

Het begrijpen hoe voetgangers zich gedragen in drukke, openbare ruimtes is in de praktijk belangrijk gebleken. Tegenwoordig worden er op grote schaal sensoren ingezet om de Wi-Fi signalen te detecteren die uitgezonden worden door smartphones. Op basis van deze waarnemingen kunnen *statistische tellingen* gemaakt worden, die inzicht geven in bezoekersaantallen, evenals hoe een menigte zich tussen verschillende locaties beweegt. Omdat de signalen van privé apparaten worden verwerkt in het proces, zijn er terechte zorgen over de privacy van de eigenaren van de apparaten. Er is gepoogd om deze zorgen weg te nemen, maar deze methoden bleken onvoldoende, grotendeels omdat het nog steeds mogelijk was om individuen te traceren.

Wij dragen twee nieuwe methodes aan die de privacy-gevoelige waarnemingen van individuen beschermen en tevens het maken van statistische tellingen toelaten. Onze eerste methode anonimiseert de waarnemingen zodra ze gemaakt zijn. De verzamelde gegevens zijn beschermd met wat we *detection k-anonymity* noemen, zelfs als de geanonimiseerde waarnemingen in de toekomst gecombineerd worden met andere data. Onze tweede methode codeert de waarnemingen in probabilistische datastructuren, genaamd *Bloom filters (BFs)*, en versleutelt de BFs met *homomorfe encryptie (HE)*. Door HE als onderdeel van de cryptografische constructie te gebruiken, kunnen de tellingen op basis van de versleutelde waarnemingen berekend worden, zonder dat de waarnemingen eerst ontcijferd moeten worden. Bovendien wordt het eindresultaat alleen bekend gemaakt bij de partij die opdracht gaf te tellen. Daarnaast, om te oordelen welke gedetecteerde apparaten, beschermd met de combinatie van BFs en HE, onderdeel van een menigte zijn, onderzoeken we de mogelijkheid om stationaire en bewegende apparaten apart te tellen, op basis van hun detectie-frequentie.

We implementeren onze voorgedragen ontwerpen, en evalueren ze uitgebreid met zowel gesimuleerde gegevens en gegevens uit de echte wereld. Onze resultaten tonen aan dat zeer nauwkeurige statistische tellingen voor voetgangersgedrag, met garanties voor privacy, mogelijk zijn.

Abstract (Romanian)

Intelegerea dinamicii multimilor in spatii publice aglomerate s-a dovedit a fi importanta. Astazi sunt instalate, pe scara larga, infrastructuri de senzori care detecteaza semnalele Wi-Fi emise de smartphone-urile oamenilor din multimi. Pornind de la aceste detectii se poate construi o intelegere a multimilor sub forma de *contorizari statistice*, oferind informatii precum numarul de oameni prezenti intr-o locatie sau fluxul multimilor intre locatii. Avand in vedere ca acest proces presupune utilizarea detectiilor dispozitivelor apartinand persoanelor, exista ingrijorari legitime cu privire la privacy-ul acelor persoane detectate. Incercarile de a rezolva aceste ingrijorari s-au dovedit insuficiente, mai ales pentru ca reidentificarea unica ramanea inca posibila.

Propunem doua noi metode pentru a proteja detectiile privacy-sensitive, pastrand in acelasi timp posibilitatea contorizarii statistice a multimilor. Prima metoda anonimizeaza detectiile pe loc, garantand protectie sub forma *detection k-anonymity* pentru toate datele colectate, indiferent cum datele anonimizate ar putea fi combinate pentru a raspunde la viitoare intrebari. A doua metoda se bazeaza pe codificarea detectiilor in structuri de date probabilistice denumite *Bloom filtre* (BF-uri), urmata de criptarea acestora cu o schema de *criptare homomorfica* (HE). Folosita in cadrul unei constructii criptografice multipartite, HE permite executarea operatiilor necesare contorizarilor statistice direct pe datele criptate, fara posibilitatea de decriptare, dezvaluind in clar doar rezultatul final si doar partii careia acesta ii este destinat. Mai mult, pentru a facilita decizii granulare cu privire la apartenenta dispozitivelor detectate la multime si sub aceleasi garantii de protectie a privacy-ului oferita de combinatia dintre BF-uri si HE, exploram posibilitatea de a contoriza separat dispozitivele nonstationare de cele stationare in functie de frecventa detectiilor acestora.

Implementam si realizam o ampla evaluare a contributiilor propuse utilizand date simulate, precum si date din lumea reala. Rezultatele noastre demonstreaza ca se pot obtine contorizari statistice de inalta acuratete pentru dinamica multimilor, in acelasi timp protectia privacy-ului fiind garantata.

Acknowledgements

When I started my PhD, I thought it was going to be a pretty standard process in which you do research according to a plan and, at a certain point, you graduate. Actually, it was not even close. Instead, I found myself in an open-ended exploration process which implied navigating towards some goals, with no clear plan to follow other than constantly reconfiguring the route as indicated by scientific discoveries made along the way and under the guidance of knowledgeable mentors. Learning how to navigate myself through this exploration process seems to have been the actual purpose. As this dissertation marks the end of my PhD studies, I believe I made it.

It was an adventurous journey. At the same time, it was a transformative one. There were many people who accompanied me, playing an important part in it. I would like to take this occasion to express my gratitude towards them.

First and foremost, I would like to thank my supervisors Maarten van Steen, Andreas Peter and Ciprian Dobre for the valuable guidance they have been giving to me throughout these years, for their dedication and for their persistence in not giving up on making a researcher out of me.

Maarten, it was both an honor and a pleasure to work together and I am deeply grateful to you for giving me this chance. Looking back now, collaborating with you had a tremendous influence on who I am as a researcher and as a person. You made me develop capabilities such as separating the essence from the noise, being precise, clear in explanations, not being fully content until the whole picture is clear, pushing boundaries, challenging myself, simplifying complicated problems until they become approachable, telling the right story, and this is definitely only a small subset of all the important things I learned from you. Also, I am thankful for all the support you gave me when I had difficulties, always with a good advice and an understanding attitude.

Andreas, most problems can have more dimensions. These dimensions can be discovered only when looking at problems from different standpoints and through different lenses. Even though I was aware of the existence of this

discovery process, I did not know how to conduct it. Now I do, thanks to you. You always brought different views on the table, thinking along with me, but also, in parallel, in paradigms that I was not initially aware of, and yet highly relevant to the problem in discussion. Does the construction stand? Is it strong, covering, robust enough? Think through, Valeriu! I will, Andreas, and thank you for ingraining this in me.

Ciprian, I am grateful to you for having been working together for the past ten years, ever since I started doing my bachelor's thesis under your supervision. From you I got a first glimpse of what research means. That glimpse was fascinating enough to make me follow this path. You actively exposed me to learning and development opportunities. At the same time, you showed me what it really means to move things forward by making them happen. You have always found the appropriate kind of support that I needed to succeed and you have offered it to me. I am thankful to you for that.

I am also grateful towards the graduation committee for taking the time to review my work and provide valuable feedback.

This story would have never been the same without the amazing colleagues I had. In or out of office, in the social corner or during lunch breaks, be they whiteboard discussions or social gatherings, I thank you for the great time spent together. For my time in Enschede, Bertine, Geert Jan and Suse, thanks for being there with spot-on support whenever I needed. Amina, Bence, Dan, Federico, Herson, Hudi, Philipp, Riccardo, Roeland, Susanne, Thijs, Tim, Yoep, we shared not only an office but also many moments worth remembering; I thank you all, it was a pleasure. Ali, Andrea, Asbat, Chakshu, Chris, Claudio, Erik, Florian, Jerre, Kemilly, Klaas, Luis, Maarten, Marten, Meikel, Una, Zsolt, as well as others that I had the chance to interact with at the UT, it was also a pleasure. My thanks also to Alex, Adrian, Catalin, Cristi, Florin, Mihai, Radu, Silviu, Tudor, with whom I spent valuable time while at UPB in Bucharest.

Adrian, Andrei and Nadia, thank you for your collaboration. Similar research interests brought us together and now you are continuing the work in the same direction as I did. I wish you all the luck.

My dear parents, Raluca and Daniel, I am grateful to you for the education you gave me, for the values you strived to instill in me, for everything you did to make me who I am today. There are not enough words to express how important your contribution was.

Teodora, you have been by my side throughout all these years. From the first to the last day, you have never stopped supporting me. You were there, understanding, encouraging and patient, no matter how good or bad the situations we had to go through were. At the same time, you brought love and happiness in my life on a daily basis. I sincerely thank you for that.

Contents

List of Figures

List of Tables

List of Articles

This dissertation is based on the following articles of the author:

1. **Stanciu, V. D.**, van Steen, M., Dobre, C., & Peter, A. (2020, December). k-Anonymous Crowd Flow Analytics. In MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (pp. 376-385).

2. **Stanciu, V. D.**, Steen, M. V., Dobre, C., & Peter, A. (2021, April). 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).

3. **Stanciu, V. D.**, van Steen, M., Dobre, C., & Peter, A. Privacy-Friendly Statistical Counting for Pedestrian Dynamics. Under review.

4. **Stanciu, V. D.**, van Steen, M., Dobre, C., & Peter, A. (2022, October). 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 on International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (pp. 213-222).

During his PhD time, the author also contributed to the following articles:

5. Rogojanu, T., Ghita, M., **Stanciu, V.**, Ciobanu, R. I., Marin, R. C., Pop, F., & Dobre, C. (2018, June). Netiot: A versatile iot platform integrating sensors and applications. In 2018 Global Internet of Things Summit (GIoTS) (pp. 1-6). IEEE.

6. Pantelimon, S. G., Rogojanu, T., Braileanu, A., **Stanciu, V. D.**, & Dobre, C. (2019, April). Towards a seamless integration of iot devices

with iot platforms using a low-code approach. In 2019 IEEE 5th World Forum on Internet of Things (WF-IoT) (pp. 566-571). IEEE.

7. Cebanov, E., Dobre, C., Gradinaru, A., Ciobanu, R. I., & **Stanciu, V. D.** (2019, June). Activity recognition for ambient assisted living using off-the-shelf motion sensing input devices. In 2019 Global IoT Summit (GIoTS) (pp. 1-6). IEEE.

8. van Steen, M., **Stanciu, V. D.**, Shafaeipour, N., Chilipirea, C., Dobre, C., Peter, A., & Wang, M. (2022). Challenges in automated measurement of pedestrian dynamics. In IFIP International Conference on Distributed Applications and Interoperable Systems (pp. 187-199). Springer, Cham.

Chapter 1

Introduction

Being able to observe the behavior of a crowd is a cornerstone for successfully managing crowded public areas. The prevalence of mobile devices paved the way for wide-scale deployments of infrastructures that perform automated sensing. Suddenly, people in a crowd could be discreetly monitored by leveraging radio signals such as Wi-Fi probe requests periodically sent by their devices. However, handling such uniquely identifying data in such a way that it does not expose the sensed individuals to potential privacy infringements proves to be a difficult task. This is the area where we position our research. Let us begin by presenting an overview of the problem.

1.1 Overview

Understanding pedestrian behavior in crowded public spaces has been a matter of interest for many years. Research within the crowd-dynamics field thoroughly explored movement patterns and different behaviors that can occur inside a crowd at different points in time [44, 55, 51]. It has already been shown that insights can be extremely valuable for urban planning [49], traffic optimization [60, 43], events organization [20, 80, 17, 82, 14], identification of travel patterns [32], uncovering social interactions [45, 27], footfall estimation [68, 42] or even public safety [46, 30, 81, 36]. Various technologies have been employed, including video cameras, mechanical counters, RFID beacons and Infrared devices. With the advent of smartphones as personal devices constantly carried by people, an enormous amount of high-accuracy information became available, foaming from inside the crowd and boasting an unprecedentedly intimate whiff, creating opportunities for automated tracking through interfaces such as

Bluetooth and Wi-Fi.

Wi-Fi-based crowd monitoring took an upper hand and became commonplace practice. Wi-Fi scanners installed in public spaces gather signals broadcasted by devices carried by individuals. By leveraging such signals, interested parties can estimate the size of crowds near those scanners, as well as the size of the flows developing between them.

Dealing with data related to crowds has always been a sensitive matter, regardless the technique used for monitoring, mainly because insights are built upon the people making up these crowds, people who have privacy concerns. In Wi-Fi-based crowd monitoring, signals gathered by scanners contain unique identifiers, i.e. MAC addresses, corresponding to devices carried by individuals. Detecting these identifiers at different locations over time allows the system to build up crowd-level knowledge on pedestrian dynamics based on the movement patterns of individuals. In such a system, an individual could be uniquely re-identified from data bearing her identifier and have her every move followed, infringing thus her privacy.

In an effort to prevent such situations from happening, sets of rules were proposed to clearly regulate the process, like, for example, the General Data Protection Regulation [63] (GDPR) in the EU. According to this regulation, the kind of information processed and stored with the intent of profiling a natural person, information which, combined with other external knowledge, could lead to uniquely identifying individuals, qualifies as personal data.

Existing data protection strategies for Wi-Fi-based crowd monitoring proved on several occasions not to provide a proper protection for the individuals being sensed [58, 28]. Currently used strategies are based on replacing real identifiers with pseudonyms obtained either by hashing the MAC addresses with a one-way hash function, encrypting them with a deterministic encryption scheme or assigning them a random token generated by a cryptographically secure pseudorandom number generator. Pseudonyms still allow tracking over time and space, as well as creating individual profiles which, under certain conditions, for example when external knowledge is available, remain susceptible to re-identification. As a result, organizations doing crowd monitoring ended up facing difficult challenges while delivering their services. Many of them halted their activities [1, 4, 5], while others are being fined due to privacy-related incidents [6]. This brings us to our main research question, that we formulate below.

1.2 Research Questions

In this thesis we aim to come up with mechanisms to protect the privacy of the sensed individuals by design while still fulfilling crowd-monitoring needs. In other words, we investigate the following *main research question*:

MRQ: How to construct Wi-Fi-based crowd-monitoring systems in such a way that they enable discovering pedestrian dynamics and protect the privacy of the sensed individuals at the same time?

To address **MRQ**, we pursue an exploration from multiple directions, materialized in three research questions. First of all, there is the privacy protection problem. Therefore, we formulate the *first research question* as follows:

RQ1: What techniques can be used for managing privacy-sensitive crowd-monitoring data such that *privacy protection* is ensured?

Crowd-monitoring data undergoes privacy protection mechanisms. This kind of processing can have consequences on the utility of the offered crowd-monitoring insights. By utility here we refer to aspects such as the accuracy of the crowd estimations, the degree of applicability of the system in certain circumstances, as well as the potential granularity of the offered insights. Thus, we come to the *second research question*:

RQ2: To what extent do the considered privacy protection techniques impact the *utility level* of the attainable crowd-monitoring insights?

To address **RQ2**, we evaluate our proposed methods on simulated data, covering a whole range of cases, as well as on real-world data to assess the impact when dealing with real detections generated by actual pedestrian movements.

Wi-Fi-based crowd-monitoring systems already process large amounts of data. Introducing mechanisms for privacy protection purposes implies an additional overhead. Such overhead has the potential to impact the practicality of the deployment. Hence, we propose the *third research question*:

RQ3: How *expensive* are the considered privacy protection techniques for crowd monitoring from an *efficiency* point of view?

Answering **RQ3** is based on performing proof-of-concept implementations of the computationally heavy operations using commodity hardware and assessing their performance when configured as being part of a crowd-monitoring system.

After presenting our work in the following chapters, we will come back in Chapter 6 with a detailed discussion on how each of the research questions has been addressed.

1.3 Outline & Contribution

The rest of the thesis is structured as follows. In Chapter 2 we present background information together with relevant related work. Chapters 3 and 4 display the two methods proposed for monitoring crowds while protecting the privacy-sensitive detections of individuals. In Chapter 5 we investigate how to separately count nonstationary from stationary devices with privacy protection enabled. Chapter 6 concludes the thesis, reflecting upon achievements, limitations and directions for future research.

Addressing the formulated research questions, we make the following contributions.

Detection k-anonymity for crowd monitoring. We propose, in Chapter 3, a novel anonymization technique that borrows from the notion of k-anonymity while avoiding its well-known drawbacks that lead to de-anonymization. The technique essentially anonymizes detected smartphones immediately at the scanner before any data on such a detection is stored for further analysis. Moreover, while ensuring what we coin *detection k-anonymity*, we also ensure high accuracy of counting when dealing with realistic pedestrian flows within crowds for which the amount of leavers and joiners is known. We evaluate the solution both in a simulated environment and in a realistic environment reproducing real-life settings.

Privacy-friendly crowd monitoring using Bloom filters & homomorphic encryption. We present, in Chapter 4, a construction that protects the short-term storage and processing of privacy-sensitive Wi-Fi detections under strong cryptographic guarantees and makes available in the clear, as end results, only statistical counts of crowds. To produce these statistical counts, we make use of *homomorphically encrypted Bloom filters* as facilitators for oblivious set membership testing under encryption. We implement the system and perform evaluation on both simulated data and a real-world crowd-monitoring dataset, demonstrating that it is feasible to achieve highly accurate statistical counts in a privacy-friendly way.

Anonymized counting of nonstationary Wi-Fi devices. Emitting Wi-Fi signals is not a feature for devices of only passersby, but also for printers, smart TVs, and other devices that exhibit stationary behavior over time,

which eventually end up affecting pedestrian crowd measurements. To deal with this problem, we propose, in Chapter 5, 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, fulfilling the same privacy-preserving guarantees as in Chapter 4. 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.

Chapter 2

Background & Related Work

Monitoring crowds of pedestrians has been a matter of study for many years, with several technologies being investigated as promising candidates. The technical capabilities of the Wi-Fi together with its inconspicuous nature of sensing people propelled it as a front runner technology. Therefore, nowadays there are numerous Wi-Fi sensing infrastructures deployed in practice across cities from around the globe. Privacy aspects, however, are not uniformly addressed, often leading to underachievements or hiding pitfalls.

In this chapter we will first present background information, showing how monitoring is possible. Then, we will provide a generic model of Wi-Fi-based crowd-monitoring system that we are going to use throughout the rest of the thesis, to understand where the privacy problem arises. Eventually, we will look at currently existing approaches aiming for privacy protection when monitoring crowds.

2.1 Background

People traveling in public spaces generally carry with them mobile devices, such as smartphones. These devices have communication interfaces that allow us to detect them when they are in the vicinity of a sensing infrastructure. Research has shown that Wi-Fi and Bluetooth interfaces [67, 20, 7] are highly appropriate for unobtrusively detecting the behavior of crowds of people. In Wi-Fi setups [56], the MAC address of the devices carried by people is detected by fixed scanners whenever they transmit probe requests meant to discover available networks. Bluetooth sensing is performed by fixed scanners that send, periodically, inquiry requests to nearby devices and then receive responses

containing the MAC addresses of the devices. For pedestrian monitoring however, Wi-Fi proved to be the better choice due to higher range, more discoverable devices and lower deployment costs [67]. For an extensive study of the matter, we refer the reader to [31].

Based on these detections, interested parties can later on derive relevant information regarding pedestrian dynamics, such as crowd densities and flows [67], as well as mobility patterns occurring within crowds [20]. The key element allowing this to happen is that probe requests are accompanied by the MAC addresses of the devices sending them, serving thus as unique identifiers in the crowd-monitoring process. At the same time, this is the main cause of concerns regarding privacy; we will come back to this later in this chapter. Let us now model a Wi-Fi-based crowd-monitoring system.

2.2 Wi-Fi-Based Crowd Monitoring

Typically, setting up a Wi-Fi crowd-monitoring system starts by installing a set of Wi-Fi scanners $\mathcal{S} = \{s_1, ..., s_n\}$ in a public space where crowds of people are expected. These scanners could be either purposefully built Wi-Fi sniffers, access points, or any other apparatus that can pick up Wi-Fi signals in their vicinity. Ideally, scanners are positioned in such a way that they have nonoverlapping ranges, to prevent them from sensing the same signals at the same time.

People carrying Wi-Fi enabled devices pass through the public space. Their devices regularly broadcast management frames called probe requests to search for available Wi-Fi networks. Probe requests are sent out in the clear and contain, along other information, the MAC address of the sender, $a \in \mathcal{A}$ where $\mathcal{A} \subset \{0,1\}^{48}$, hereby acting as a unique identifier for the broadcasting device. Whenever such a device is identified by a scanner as passing through its range (i.e. a probe request is received from it), the scanner learns its MAC address a as well as the timestamp of reception t, and it associates it with an epoch $e \in \mathcal{E}$ such that $t_{start}(e) \leq t < t_{end}(e)$, where t_{start} and t_{end} mark the beginning and the end of an epoch and $\mathcal{E}$ denotes the set of all such epochs. We call the 3-tuple (a, s, e) a *detection*, signifying that a device with MAC address a was detected by scanner s during epoch e.

We model a crowd as a set of detections $\mathcal{D}_{s,e}$ containing the devices detected by a scanner s during an epoch e. This decision implies two effects. First, using a set ensures that a device is counted by a scanner only once per epoch even if it may broadcast multiple probe requests. Second, we consider the number of detected devices as the number of people, this being the only information such

a system can gather. We are aware that the actual number of people may be different (e.g., due to some people not carrying mobile devices) and we assume that a correction factor (e.g., how it is proposed in works such as [69]) will be applied afterwards.

To address incoming crowd-monitoring queries, detection sets are usually stored on a central server. Detection sets contain MAC addresses of devices belonging to individuals, representing thus privacy-sensitive information. Handling such unique identifiers must be done carefully, as it was shown that unconsented tracking [26] or even profiling [27] may happen otherwise. Therefore, privacy protection measures must be employed.

2.3 Privacy Protection Directions

Being able to collect precise information on the whereabouts of individuals without any explicit consent is a major privacy problem. This has been investigated by hardware manufacturers, by organizations having crowd-monitoring deployments and also by researchers in this field.

Hardware manufacturers tried to address the privacy issue by implementing MAC address randomization, a mechanism that replaces the real MAC addresses with random ones when sending out probe requests. Despite sometimes being effective, it proved to be insufficient, as works such as [78] and [53] showed, re-identification remaining possible through several techniques mostly because of unclean and inconsistent deployments across the wide range of manufacturers, which still remains a problem as of 2021 [40]. Leveraging this, there are works [77] that show how different randomized MAC addresses belonging to the same physical device can be clustered together and treated as a unique device for crowd-monitoring purposes. Besides that, devices use their real MAC addresses when they are connected to a network, a situation in which randomization is of no use.

Wi-Fi crowd-monitoring organizations tried to address the problem as well. The prevalent approach employs pseudonymization, a technique that replaces too the original MAC addresses in probe requests, this time on the capturing side. So-called pseudonyms result after applying either a one-way hash function, a randomized allocation or a deterministic encryption scheme on the original MAC addresses. However, the MAC address space is limited to 2^{48}, so any resulting pseudonyms are susceptible to brute-force attacks, as they are known as weak anonymized data [29]. Furthermore, it was shown by Demir et al. how most commercial solutions using such mechanisms can be broken using off-the-shelf equipment [28], also reconfirmed by Marx et al. [54] in a more

recent work. This comes on top of the fact that a survey by Draghici and van Steen [31] discovered that not using privacy-protecting methods is far from being a rare event.

There are researchers who looked specifically into protecting the privacy of individuals sensed by Wi-Fi crowd-monitoring systems. There are three main directions that show potential here.

A first approach that comes to mind when thinking about protecting the privacy of individuals whose data is stored in a dataset while still providing useful statistical information based on it, is *differential privacy* [33]. In such setups, to achieve privacy protection, data is usually perturbed with noise [34]; to produce the same amount of privacy for queries involving a decreasing number of individuals, more noise needs to be added. A notable effort towards privacy-preserving crowd monitoring using differential privacy is presented by Allagan et al. [11]. The authors make use of differentially pan-private Bloom filters (BLIPs), i.e. probabilistic data structures in which they store the data and perturb it for privacy protection. While their method works well for large crowds, it achieves low accuracies when handling small crowds, this being a common condition of systems implementing differential privacy.

A second direction pursued by researchers is striving for privacy through anonymity and, in particular, *k-anonymity* [66], which is the kind of protection achieved when the information about a person contained in a release is indistinguishable from k-1 other persons. For crowd-monitoring systems, Kamp et al. [47] proposed a protection mechanism based on linear counting sketches [65]. Their approach allows estimating footfall in one location as well as reconstructing crowd flows between different locations, privacy protection being based on *expected* k-anonymity over the identifiers, which comes as a natural property of sketches. In contrast, also building around k-anonymity, our work presented in Chapter 3 introduces *detection k-anonymity*, an anonymity measure which is, this time, *guaranteed* by an active mechanism for footfall and crowd flow queries while eventually allowing estimations of the concerned crowds. There we will also present a broader background on anonymity and k-anonymity.

Finally, privacy-sensitive data of individuals can be protected through *cryptographic* means. In such constructions, once encryption has been applied on the privacy-sensitive data, one cannot go back to its original form under the assumption that a set of security requirements is followed. Moreover, by using, for example, a homomorphic encryption scheme, mathematical operations can be performed on the data without the need for decryption, making it a promising candidate for privacy-preserving crowd monitoring under encryption. We design such a construction to support a crowd-monitoring system in Chapter 4, that we

also use later in Chapter 5 for separately counting nonstationary from stationary devices. In these chapters we will provide more background information on the matter.

Chapter 3

Detection k-Anonymity

In this chapter, we propose a novel privacy protection method for crowd monitoring that preserves the privacy of all monitored individuals under anonymity guarantees while maintaining high accuracy of measurements. Our mechanism leverages k-anonymity principles on top of truncated identifiers, dropping the usage of unique identifiers and ensuring, for any formation of crowd-monitoring scenarios, that there is no individual having her privacy compromised. Moreover, the mechanism is computationally lightweight, running in linear time, and can be applied in a live manner right at the collection point even before the sensing data reaches the crowd-monitoring database, thus complying with requirements of anonymization on the fly. We evaluate our construction both in a simulated environment, to test edge cases and behavior when ranging different parameters, and in a realistic environment reproducing real-life settings.

This chapter is based on the work presented in [72]. The rest of the chapter is organized as follows. Section 3.1 presents the system model, together with the theoretical grounds supporting our construction. Section 3.2 introduces the experimental setup, the metrics used and the employed mechanisms, while in Section 3.3 a thorough evaluation is performed. In Section 3.4 a review of related literature is provided and then, finally, Section 3.5 concludes the chapter.

3.1 System model

3.1.1 Overview

Crowd monitoring is a process usually performed to get insights regarding crowds of people, such as discovering either the presence or movement patterns

happening inside a certain public or private environment [83]. Regardless of the technology used for sensing (e.g., Wi-Fi or Bluetooth scanners, video cameras, and so on), it relies on detecting people passing by several collection points at different time intervals. For example, in the case of Wi-Fi, a mobile device regularly broadcasts probe requests containing its MAC address as a unique identifier, which can be subsequently picked up at a Wi-Fi scanner. In a naïve setting, a device detection is constructed at the scanner as a triplet containing a device's MAC address, a timestamp, and the scanner's identifier. Such triplets are stored in a central database for further analysis.

Clearly, without taking further measures, privacy infringement is at stake. More specifically, we consider a threat model in which an attacker has access to the stored detections. By making use of these detections and assuming no additional background knowledge, the attacker aims to uniquely trace back to the individuals who generated them. We do not consider other attackers, such as attackers manipulating the sensed data or injecting false detections.

As an advancement of state-of-the-art methods, we propose to perform a novel anonymization process on the fly, directly at the scanner, or more general at collection points, before detections reach the server, a process that we will introduce later on in this chapter. For clarity and without loss of generality, we will assume that Wi-Fi sensors are used.

In our construction, when we talk about movement patterns of crowds, we specifically refer to being able to understand pedestrian crowd flows, i.e. how people constituted in a crowd circulate through public spaces. To achieve this, we need to build our system in such a way that it offers high accuracy of measurements for this kind of scenarios while offering anonymity guarantees for all the data being stored.

3.1.2 Formalities

Let us now present the formalities that we are going to use in this chapter. We recall that a Wi-Fi crowd-monitoring environment, as we define it in our construction, consists of:

- A set $\mathcal{S}$ of N scanners, which could be either access points, Wi-Fi sniffers, or any other device able to gather Wi-Fi messages. We make the assumption that scanners have nonoverlapping ranges and they run the protocol as expected.

- A set $\mathcal{E}$ of K epochs during which the system runs; the duration of the epochs is established according to the specificity of the environment. We

assume that each epoch lasts τ time units. $T = K \cdot \tau$ is the total time span during which we perform crowd-monitoring activities.

- A set IDS of M people being detected throughout our system during T; each person is represented by a unique 48-bit identifier, be it a MAC address or other pseudonym.

We consider a detection as a triplet $(id, s, e), id \in IDS, s \in \mathcal{S}, e \in \mathcal{E}$, representing a person uniquely identified by id, sensed by scanner s during epoch e. Let $D(s, e) \subset IDS$ be the set of identifiers detected at scanner $s \in \mathcal{S}$ during epoch $e \in \mathcal{E}$. In our system we assume that, at the end of each epoch, the detections collected by the scanners undergo an *anonymization process* P:

Definition 1. Let 2^{IDS} denote the powerset of the set IDS. We define an **anonymization process** $\boldsymbol{P}$ as an algorithm $P : 2^{IDS} \times IDS \longrightarrow PIDS$ that takes a set of identifiers $A \in 2^{IDS}$ and an identifier $id \in A$ as input and outputs an anonymized identifier $pid \in PIDS$, where $PIDS$ denotes the set of all possible identifiers that are anonymized with respect to P, including the special symbol $\perp$ (which captures the "removal" of identifiers for anonymization).

By modelling P to take as input both an identifier id as well as an identifier set A in which id resides, we enable P to anonymize id depending on its "environment" A. To ease readability, for $id \in A \in 2^{IDS}$ we write $P(A, id)$ simply as $P_A(id)$ or even as $P(id)$ if there is no ambiguity about the underlying set A. For any $B \subseteq A$, we interpret $P(A, B)$ as $\bigcup_{b \in B} P_A(b)$. We note that for a set $A \in 2^{IDS}$, $P_A(A)$ defines a multiset for which $m(pid) = |\{j \in A \mid P_A(j) = pid\}|$ denotes the multiplicity of $pid \in P_A(A) \setminus \{\perp\}$[1]. The multiplicity $m(\perp)$ of $\perp$ in $P(A)$ is always set to 1 (as removed identifiers are assumed to be nonreconstructable).

Notation. For a detection set $D(s, e) \subseteq IDS$ and anonymization process P, we denote the multiset $P_{D(s,e)}(D(s, e))$ as $PD(s, e)$.

A simple example of such an anonymization process P is the truncation operation $trunc(id, nb)$ which removes all but the last nb bits from the binary number id, i.e. $trunc(id, nb) = id \mod 2^{nb}$, for all $id \in IDS$. In this example, $IDS \subseteq \{0, 1\}^{48}$ while $PIDS = \{0, 1\}^{nb}$. We will see more examples of anonymization processes P later in the chapter.

After undergoing the process P, detections are stored as multisets in a database. The purpose of this *crowd-monitoring database* (*CMD*) is to provide

[1] We write $m_A(pid)$ instead of $m(pid)$ when the context is ambiguous.

meaningful answers to *crowd-monitoring queries*. Those queries are modelled again as multisets.

Definition 2. For scanners from $\mathcal{S}$ and epochs from $\mathcal{E}$ we define a **crowd-monitoring query** as a multiset $PD(s, e)$ and any AND-combinations of such multisets. In particular, a **simple query** is a single multiset $PD(s, e)$ for some $s \in \mathcal{S}$ and $e \in \mathcal{E}$. A **composite query** CD is an AND-combination of multisets over a collection $\mathcal{D} = \{PD(s, e)\}$ and is defined as the multiset

$$\{pid^m | pid \in \bigcup_{d \in \mathcal{D}} d; m = \min_d \{m_d(pid)\}\}.$$

Composite queries, as they are defined above, cover a broad spectrum of situations; many of these situations are not relevant for crowd analytics, while some are even impossible (such as detecting the same device at different locations at the same time). As we mentioned, we are interested in composite queries regarding *crowd flows*. Envisioning crowd flows, we expect to encounter people detected as moving together in the form of a crowd between different scanners over time. Under ideal circumstances, a crowd identified as being together at a certain point should be also detected in its entirety as it travels. However, in reality there are people leaving as well as joining a crowd flow, thus creating variations of *ideal crowd flows*. Ideal crowd flows and their variations form the focus of our investigations.

Definition 3. An **ideal crowd flow** (of size n) is a collection of detection sets $\mathcal{D} = \{D(s_1, e_1), \ldots, D(s_n, e_n)\}$ where $e_i < e_j$ for $i < j$, such that $\bigcap D(s_j, e_j) \in \mathcal{D}$. We call this situation "ideal" because in one of its detection sets we capture a crowd which is also *fully encountered* across all the other detection sets. Such an ideal crowd flow is depicted in Fig. 3.1.

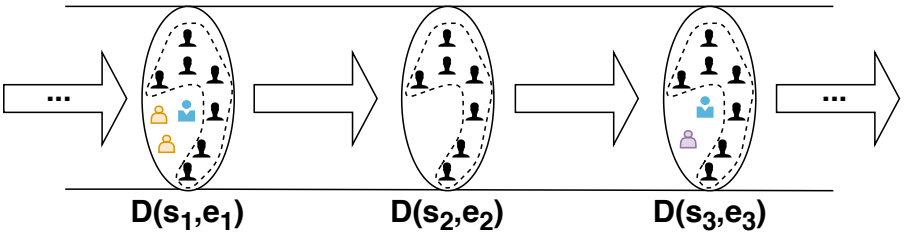


Figure 3.1: Example of ideal crowd flow of size 3.

Definition 4. Let *CF* denote an *ideal crowd flow* of size n. Let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{n-1}\}$ be a set of percentages, where λ_i represents the percentage of devices that have left *CF* during e_i (i.e. these devices were detected during e_i, but no longer during e_{i+1}). Likewise, let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{n-1}\}$ be a set of percentages, where γ_i represents the percentage of devices that joined *CF* during e_i, meaning that these devices were detected during e_i, but not during e_{i-1}. We define such a flow as a ***(Λ, Γ)-crowd flow***. We display an example in Fig. 3.2.

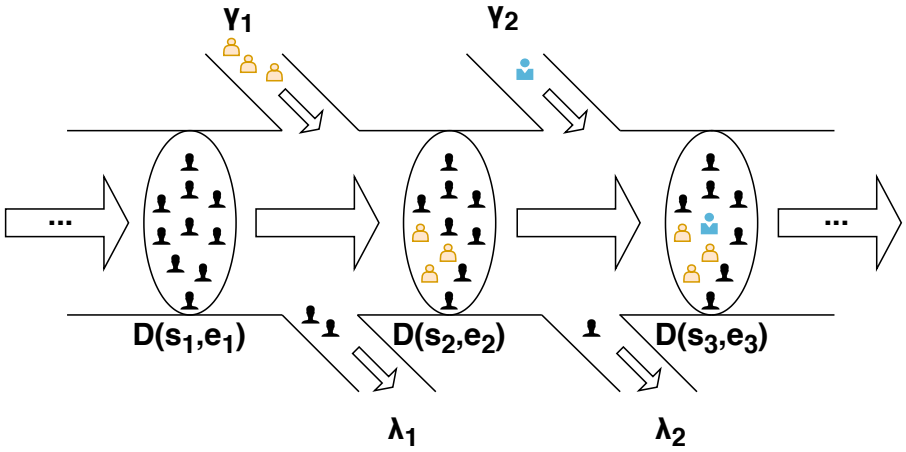


Figure 3.2: Example of (Λ, Γ)-crowd flow of size 3.

3.2 k-Anonymous Crowd Flow Analytics

3.2.1 Metrics

We have shown in the previous section how *CMD* is built and what kind of crowd-monitoring queries are to be performed onto it. Now we focus on how to assess the effectiveness of a given anonymization process P in protecting the anonymity of individuals, as well as to measure its impact on the quality of outcomes expected from the system.

In terms of anonymity, we adapt the widely used notion of *k-anonymity* [66] to our setting of detection sets and introduce the notion *detection k-anonymity*.

Definition 5. We call an anonymization process P **detection k-anonymous** if $\forall A \subseteq IDS, id \in A : m(P_A(id)) \geq k$ or $P_A(id) = \perp$.

An anonymized identifier *pid* should correspond to at least k identifiers from the original set. Note that for sets smaller than k, the only option is to transform each identifier to $\perp$, since there are not enough identifiers in the original set to proceed differently.

Applying such a process on all the detection sets at collection points generates, by construction, multisets that can yield answers only to *detection k-anonymous simple queries*. We will show that this is sufficient in order to protect the anonymity of individuals under detection k-anonymity guarantees for any crowd-monitoring query, be it simple or composite, as it also exclusively leads to *detection k-anonymous composite queries*.

Definition 6. A (simple or composite) query CD is said to be **detection k-anonymous** if $\forall pid \in CD : m(pid) \geq k$.

Theorem 7. Consider a collection of detection k-anonymous simple queries $\mathcal{D} = \{PD(s, e)\}$ over a set of scanners $\mathcal{S}$ and epochs $\mathcal{E}$. The composite query CD obtained by composition over these simple queries is also detection k-anonymous.

Proof. Consider an identifier $pid \in CD$. By definition of a composite query, we know that $m_{CD}(pid) = \min\{m_{PD}(pid)\}$ for any $PD \in \mathcal{D}$ for which $pid \in PD$. As each $PD \in \mathcal{D}$ is detection k-anonymous, we have that $m_{PD}(pid) \geq k$, and thus also $m_{CD}(pid) \geq k$. □

Anticipating further discussions on re-identification, apart from detection k-anonymity, an anonymization process is under scrutiny regarding its *l-surjectiveness*, as defined below.

Definition 8. We call an anonymization process P **l-surjective** if $\forall id \in IDS : m(P_{IDS}(id)) \geq l$.

When P is applied on the entire IDS, the resulting multiplicities represent the *actual* number of *physical* devices behind each anonymized identifier in the dataset. Therefore, in other words, an l-surjective anonymization process ensures that any resulting anonymized identifier is shared by at least l real devices in *CMD*, no matter the query. Imagine a trivial process which simply takes a query and makes each identifier in it occur k times. Despite detection k-anonymity being respected, an attacker can immediately trace back to individuals with a probability of 1. To avoid such a situation, l-surjectivity acts

as a fallback solution, because the attacker can only guess correctly with a probability of $1/l$. Hence, it is highly important to choose the parameters of the crowd-monitoring system such that they lead to a satisfactory value of l, i.e. $l \gg k$.

Besides measuring the anonymity achieved by individuals, we are interested in the impact on the quality of outcomes expected from the system. Hence, we need to introduce an *accuracy* metric to express how far the answers to crowd-monitoring queries are from their original values after applying the anonymization process.

Definition 9. Let CD be a simple or composite crowd-monitoring query. Then, with $CD^* = CD \cup \{\perp\}$, let $IDS(CD^*)$ denote the identifiers in IDS detected by the scanners, as they were before applying the anonymization process P. We define the **query accuracy** as follows:

$$Acc(CD) = 1 - \frac{Abs(|CD| - |IDS(CD^*)|)}{|IDS(CD^*)|}$$

Abs denotes the absolute value. Special situation: if there was no identifier detected, respectively $|IDS(CD^*)| = 0$, then $Acc(CD) = 1$.

Anonymization could remove identifiers by transforming them to $\perp$, while manipulating the remaining ones together with their occurrences. Recalling this, what Definition 9 actually captures is the relation between the number of anonymized identifiers obtained as answer to the crowd-monitoring query and the number of original, nonanonymized identifiers, as they were before applying any anonymization process.

3.2.2 Mechanisms

We have as main goals achieving high accuracy for the kind of crowd-monitoring scenarios that we are interested in, as well as preserving the anonymity of all the individuals under detection k-anonymity guarantees. Let us then proceed on a quest addressing, as layers, different mechanisms needed for fulfilling these requirements.

In our system, we perform anonymization at scanner level on an epoch basis. We are willing to manipulate the detection sets in such a way that they deem detection k-anonymous simple queries. This is equivalent to saying that after applying anonymization, for each *id* from an input set $D(s, e)$, its associated *pid* should occur at least k times in an output multiset $PD(s, e)$. Following a layered approach, we chain several mechanisms, each of them representing an

anonymization process by itself but inflicting changes only to the *pid*'s that have both not yet been manipulated to occur at least k times, but also not ending up to $\perp$.

Pseudonymization, a de-facto standard found both in literature and industry, represents a flavour of an anonymization process P, as it adheres to Definition 1. However, it is a weak mechanism since it simply does a one-to-one mapping of identifiers, leaving no way for k-anonymity aspirations. Nevertheless, it is important to apply it as a first step because it strips the identifiers from any connection with their original meaning.

Applied as a second layer on top of pseudonymization, the previously introduced truncation operation $trunc(id, nb)$ has the potential to achieve better results in terms of anonymization. It can lead to a many-to-one mapping of identifiers if the number of bits to truncate is well chosen, in accordance with the size of the detection sets. While, regardless of the number of bits being truncated, the accuracy of simple queries cannot be affected (resulting multisets have the same sizes as the original sets), the situation is different for crowd flows as we discovered through experiments. To illustrate, we display in Fig. 3.3 the results of an example experiment concerning a $(\{30\}, \{50\})$-*crowd flow* (i.e. 30% leavers, 50% joiners) containing 1000 identifiers and a desired anonymity of k=2. On the y-axis we show both the accuracy and the inherent k-anonymity achieved when ranging nb as displayed on the x-axis. By inherent k-anonymity we mean the fraction of people in a crowd flow that have their corresponding truncated identifier occurring at least k times after applying the truncation alone. When the parameters are chosen in such a way that the crowd-monitoring query gets close to being detection k-anonymous, the accuracy is the lowest. The accuracy gets higher when the inherent k-anonymity decreases and, as a consequence, the query gets farther from being detection k-anonymous. The inflection points of the curves can slide left- or rightwards when Λ, Γ, k or when the crowd size are changed, but the pattern remains the same. Besides that, the truncation operation, as an anonymization process, cannot be even by definition detection k-anonymous because it does not work for settings in which $A \subseteq IDS, |A| < k$.

Building on top of the detection k-anonymity inherently obtained through truncation, to preserve the high accuracy of queries and, at the same time, resolve the remaining nonanonymized individuals, we present, as a third layer, a *correction mechanism*. A simple method would be to drop the truncated identifiers corresponding to nonanonymized individuals, but this would dramatically lower the accuracies. The same holds for another method at hand, which is inserting copies until each truncated identifier reaches at least k multiplicity. We hypothesize that using a smart combination of consistently adding copies

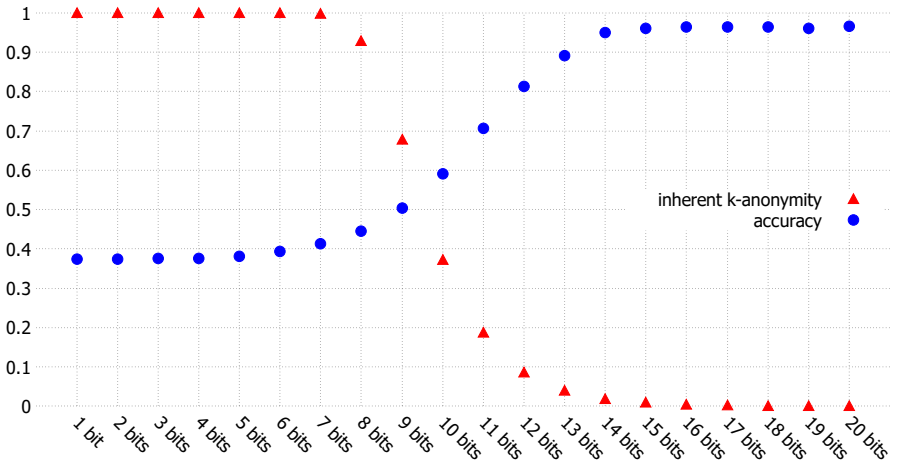


Figure 3.3: Detection k-anonymity versus accuracy when performing truncation on a $(\{30\},\{50\})$-*crowd flow* with 1.000 identifiers, k=2, nb ranges from 1 to 20.

or removing identifiers has a minimum impact on the accuracy.

Definition 10. Let us suppose that we apply a truncation operation keeping nb bits and let $CIDS(nb,k) \subset PIDS$ be a set of identifiers such that $|CIDS(nb,k)| = 2^{nb}/k$. For a resulting simple crowd-monitoring scenario PD we define a **correction mechanism** as a transformation $T : PIDS \longrightarrow PIDS, T(PD) = PD^*$, such that $\forall pid \in PD$,

$$m_{PD^*}(pid) = \begin{cases} m_{PD}(pid), \text{ if } m_{PD}(pid) \geq k \\ k, \text{ if } m_{PD}(pid) < k \ AND \ pid \in CIDS(nb,k) \\ 0, \text{ if } m_{PD}(pid) < k \ AND \ pid \notin CIDS(nb,k) \end{cases}$$

The correction mechanism, as we can see, affects only part of the identifiers: the nonanonymized ones. If we assume a uniform distribution of identifiers at query level, the mathematical expectation (when given enough queries; cf. law of large numbers) is that the inserted identifiers will perfectly balance the removed ones. At the same time, the mathematical expectation is that when composing simple queries into ideal crowd flows, the count of identifiers originally present in the intersection and removed by the mechanism to be equal to the count of the ones present in the intersection after being inserted

for detection k-anonymity purposes, thus not affecting the accuracy at all. In reality, though, there will be some limited changes in the accuracy, which we measure through experiments as discussed in Section 3.3. There are two reasons why accuracy is affected. First, although we can ensure uniform distribution of identifiers globally by, for example, using a uniformly distributed hash function as pseudonymization mechanism, there is no way we can guarantee such uniformity at query level. Second, in reality we encounter (Λ, Γ)-crowd flows rather than ideal crowd flows.

A detailed description of the actual implementation of our entire detection k-anonymous anonymization process is presented in Algorithm 1. The algorithm works with any pseudonymization mechanism of choice, be it hashing, tokenization or other method. As a correction mechanism, we use a best-effort adaptation of Definition 10, which, under the assumption of simple query-level uniformly distributed identifiers, is identical with the original one, but in practice, when the assumption does not hold, it takes care that the accuracy of simple queries is not severely affected. Essentially, instead of fixing $CIDS$, for example, to a uniform random sample of size $(1/k)$-th of the original pseudonyms space, we systematically look at the IDs that violate detection k-anonymity, order them, and keep only the first $(1/k)$-th part.

3.3 Evaluation

The anonymization process that we have introduced as a composition of three different mechanisms is detection k-anonymous, protecting the anonymity of individuals for any crowd-monitoring query. In this section we analyze the impact of applying this process on the accuracy of the (Λ, Γ)-crowd flows.

3.3.1 Simulated environment

To get a clear understanding of the behavior of our anonymization process, in our evaluation we generate detections to emulate the scenarios that we are interested in. By doing this we can freely test our design in numerous settings and we can focus on the process itself as a theoretical construction. There are a number of parameters that shape the experiments, in particular those related to physical settings (size of the crowd, number of epochs, percentages of leavers - Λ, percentages of joiners - Γ) and those related to the anonymization process (values of k, truncation nb parameter).

For the simplicity of the exposition and easiness of interpretation, we start by looking at Λ and Γ of length 1, representing crowd flows traveling from one

```
Input: DSE[ ] //Detections made by a scanner during an epoch;
Input: nb //Number of bits to keep;
Input: k //Desired value for k;
Output: PDSE[ ] //Anonymized detections;

TIDS := [ ];

foreach DSE as currentId do
    /* Compute the pseudonym of each identifier */
    pseudoId := computePseudonym(currentId);

    /* Apply the truncation operation */
    truncId := trunc(pseudoId, nb);

    if containsKey(TIDS, truncId) then
        count := getValue(TIDS, truncId);
        updateValue(TIDS, truncId, count+1);
    else
        addKeyValue(TIDS, truncId, 1);
    end
end

/* Apply correction to reach detection k-anonymity */

PDSE := [ ];
breakingPids := [ ]; //pids disobeying detection k-anonymity

foreach TIDS as (pid, count) do
    if count ≥ k then
        addCountCopies(PDSE, pid);
    else
        totalBreaking += count;
        add(breakingPids, pid);
    end
end

sortAscending(breakingPids);
breakingToKeep := floor(totalBreaking/k);
for i := 0 to breakingToKeep do
    addKCopies(PDSE, breakingPids[i]);
end

return PDSE;
```

Algorithm 1: Our anonymization process.

scanner to another between two epochs. We assume to initially have a crowd of 1000 people; part of it travels to the next scanner, part of it leaves the flow,

while new people join the flow on the way. The number of bits to keep nb depends on k and on the expected sizes of the crowd. For example, in case of a crowd of 1000 uniformly distributed identifiers, as previously shown in Fig. 3.3, 7 bits ensure almost full inherent 2-anonymity but deem low accuracies; 11 bits still ensure some degree of inherent anonymity even for k equals 3 or 4, but with much higher accuracies. Thus, we fix nb to 11. We do not fix k though, since an acceptable value can be only decided when building the crowd-monitoring system, as part of the design process; instead we run experiments for different values. We intend to see the accuracy when the percentages of joiners and leavers vary. Initially, we wanted to show it as a single figure, with both joiners and leavers fluctuating, but instead, for a much easier interpretation, we decided to split it in two. Therefore, we first show what happens when the percentage of leavers varies and the percentage of joiners remains constant, then we look at the case in which the leavers are fixed and the joiners fluctuate. Afterwards, we also clarify what happens when faced with different percentages of joiners in the first and leavers in second case.

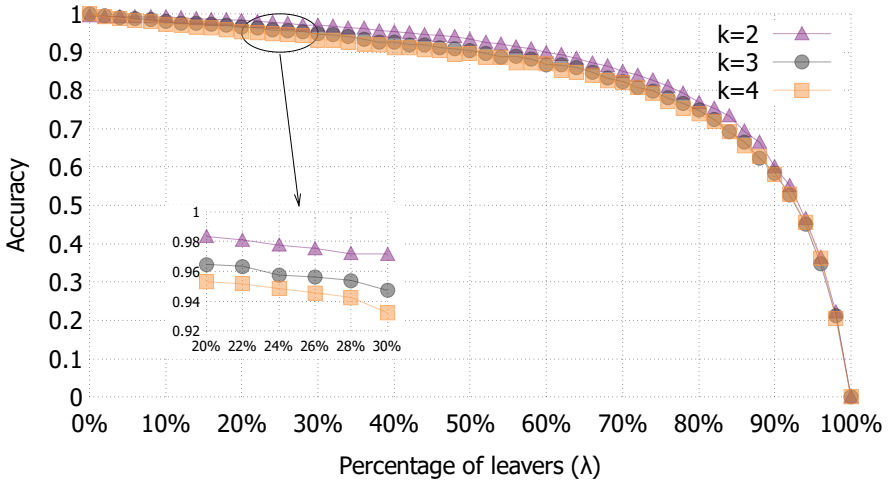


Figure 3.4: Crowd flows accuracy, γ=20, λ ranges from 0 to 100.

In the first experiment we assume that the flow starts with a crowd of 1000 people and, before reaching a second scanner, a fixed number of 200 new people join the flow. In Fig. 3.4 we display the accuracy of the corresponding composite query having our detection k-anonymous process in action when the percentage of people leaving the flow ranges from 0 to 100%. For each pair

(Λ, Γ) we perform 100 simulation runs, for each run generating a different set of uniformly distributed identifiers, and we display the mean accuracy of those runs. Intuitively, the accuracy of queries decreases when the fraction of leavers increases. It slowly decreases as long as there are enough people remaining in the crowd flow; it abruptly decreases when there are more people joining the flow than remaining in it, but at that point we consider that we are not looking at a realistic crowd flow any more since we are already dealing with different crowds mixing together. For the configurations in which the percentage of leavers is lower than 70% the system achieves accuracies higher than 0.8 for all the tested values of k. Note, however, that for higher desired values of k the truncation operation should decrease the number of bits to be kept. The reason for this is to avoid ending up with each truncated identifier occurring k times only because, at that point, a privacy attacker can try guessing with a $1/l$ probability (as l-surjectiveness indicates) who is behind an anonymized identifier.

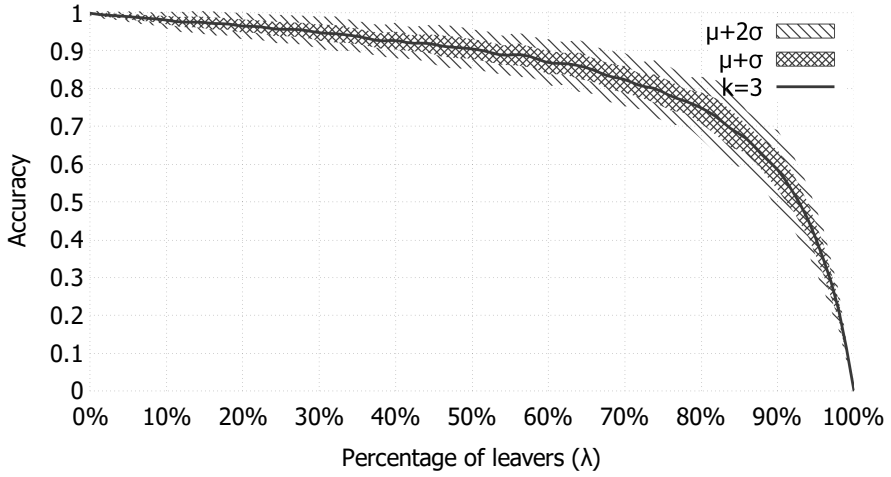


Figure 3.5: Standard deviations of crowd flows accuracy, γ=20, λ ranges from 0 to 100.

For the same settings as above, we plot, for each configuration, the standard deviations within the 100 simulated runs. For clarity reasons, we choose to show this graph separately for one specific value of k; the graph looks similar for other values of k as well. 68% of the accuracy values are within the dotted surface while the striped surface covers 95% of them. Thus, as we can see, they

are close together, the standard deviation ranging between 0.003 (0% leavers) and 0.052 (90% leavers).

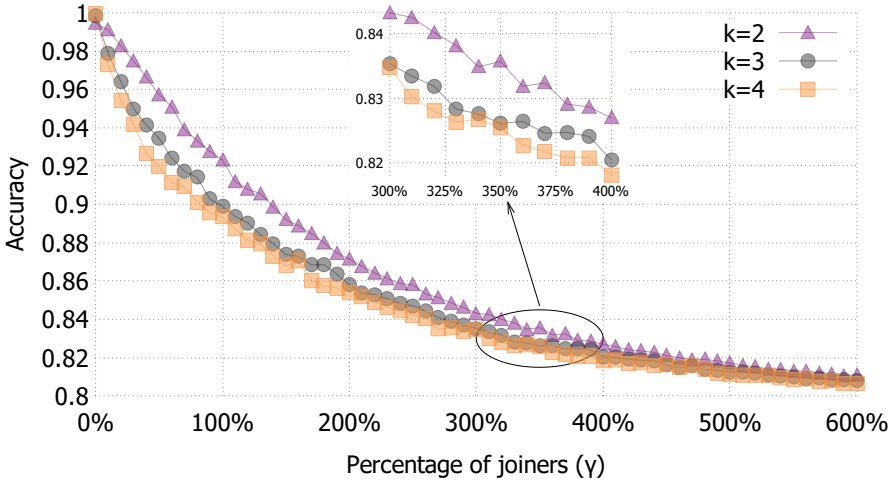


Figure 3.6: Crowd flows accuracy, λ=20, γ ranges from 0 to 600.

A second experiment concerns situations in which the percentage of people leaving the flow remains constant, while the number of people joining the flow increases. Again we start with a crowd of 1000 people, a fixed percentage of 20% of them leave the crowd before reaching a second scanner and between 0 and 600% new people join; we perform 100 runs for each (Λ, Γ) pair, for each run generating a different set of uniformly distributed identifiers, exactly as we did in the first experiment, and we display the mean accuracy of those runs. The results are displayed in Fig. 3.6. We interpret them as follows: the accuracy stays above 0.9 as long as the remaining crowd is larger than the number of new people joining the flow and it can only go as low as 0.8 (note the y-axis) when there are 6 times more people joining the crowd than they were originally in it (in our case, 6000 new people joining). This lower bound has a theoretical explanation, being dictated by the value of Λ. In a worst-case scenario, the number of actual leavers may go completely undetected. This can happen when among the increasing number of joiners there are, after anonymization, enough identifiers to match all of the 200 leaving persons. Calculated according to the accuracy formula, the theoretical lower bound in this case is 0.75. Lastly, it is worth mentioning that fixing the percentage of leavers to a different value will generate a similarly-shaped graph, but with a different slope; the same holds

for the first experiment when fixing, this time, the percentage of joiners to a different value.

Now that we understand how our detection k-anonymous process influences the accuracies of crowd-monitoring queries when applied to simple (Λ, Γ)-crowd flows, let us move forward and investigate realistic scenarios from a well-known real-life deployment.

3.3.2 Reproducing real-life deployment settings

To get insights on how people use public underground transport and to explore potential improvements, in 2016 Transport for London conducted a pilot Wi-Fi data collection experiment across 54 London Underground stations [2], publishing their findings in a detailed review [3]. Salted hashing of MAC addresses was used as a privacy-preservation mechanism, a typical example of pseudonymization bearing all the pitfalls mentioned in the introduction. Starting with July 2019, data collection is performed across the whole London Underground network, using tokenization (i.e. the assignment of a unique random value to each MAC address) instead of hashing. Considering that tokenization does not solve the previously presented issues of pseudonymization, we investigate what impact our anonymization process has on their results, arguing that our solution can be successfully applied in such settings, evolving from pseudonymization to anonymization.

The experimental Wi-Fi crowd-monitoring environment, in this case, consists of a set $\mathcal{S}$ of 1070 scanners distributed across the 54 stations, a set $\mathcal{E}$ of epochs covering the total timespan T of the experiment (from 21 November to 19 December 2016) and a set IDS of 5.6 million devices detected by any of the 1070 scanners during the experiment. Choosing, for example, the epoch length τ as 1 minute would mean that the total number of epochs in this experiment is 41760. Then, fixing the number nb of bits to be kept to 11 and running 100 experiments concerning 5.6 million uniformly distributed identifiers, we could see that, on average, at least l identifiers map to each anonymized identifier, with $l = 2559$. The main concern of the study, besides looking at statistical values and measurements, was to visualize the real flows of people inside the Underground network, to see the specific routes that are chosen between a source and a destination station, to measure train-level congestion and crowdedness. This is equivalent with having a look at the devices detected by a number of scanners in a sequence of epochs representing a journey, successfully mapping to the *crowd flows* introduced in this chapter.

As building blocks, we need to correctly identify scanner-epoch combinations in order to be able to spot the devices carried by persons taking a specific train,

as well as the devices carried by persons who get off a train. By doing this, we are able to model the whole range of situations, i.e. the start of a journey, intermediate connections, and the end of a journey. Assuming that the train schedules are known, the solution for identifying the size of the crowd making a journey on a specific route should take into account the detections made in the following settings:

- Scanner s_s on the platform of the origin station, epoch e_s before a train arrives
- Scanners on the platforms of the connecting stations, epochs before the intermediate trains arrive
- Scanner s_d on the platform of the destination station, epoch e_d after the train of that journey has completely departed from the destination station

Some could argue that the assumption that the devices are indeed detected within the specified epochs is unrealistic due to heterogeneous crowd dynamics or sensing technology limitations. That does not affect our argumentation though since it is not related to our anonymization process; this has to do with the baseline functioning of the crowd-monitoring system itself, which is a distinct problem.

The London Underground Network has some particularities making us claim that most of the journeys can be uniquely modelled through two-epoch crowd flows, regardless of the source, destination or number of connecting stations. Looking at, for example, all the 17 routes between King's Cross St. Pancras and Waterloo, as they are presented in the published review [3], one can immediately see that 12 of them have unique combinations of source and destination platforms. This means that in this case, for each source platform s and destination platform d, it is enough to simply look at the detections made by scanners s_s and s_d during the epochs matching the beginning and the end of the analyzed journey. Detections made at intermediary stations are not needed for shaping the crowd flow since the routes are already unique by source and destination. The remaining five routes have the same combinations of source and destination platforms, but contain different connecting stations. Even if it seems rare, we could encounter the following situation: some people begin a journey on the same train, they then take different paths at some point, and then they end up, again, on the same train, arriving together at the destination. To model these alternative paths, three-epoch crowd flows are needed. Please note, however, that if, according to the circulation schedule, the alternative

paths cannot lead to boarding on the same final train, a two-epoch crowd flow is still sufficient for modelling even these granular routes.

The accuracy of the crowd-monitoring queries related to (Λ, Γ)-crowd flows is highly influenced by the percentages of leavers and joiners. In the current environment, the leavers are those detected during e_s but remaining on the platform after the source train departs (we can assume that they are waiting for another train), plus the ones taking the source train and going to a different destination than the one that we are looking at. The joiners are the persons detected on the platform after the destination train has left and were either there before the train arrived (we can assume, again, that they are waiting for another train) or came by train but from another source station than the one that we are looking at. We already know from previous experiments that our anonymization process performs well in terms of accuracy when the leavers and joiners are not overwhelmingly high relative to the size of the crowd, indicating popular routes as candidates for high accuracies. We can then immediately see that crowd flows originating or ending on platforms which have a unique line passing through them have a higher chance of achieving high accuracy. Since there are no trains going somewhere else to be waited for, the leavers and joiners would be at a minimum.

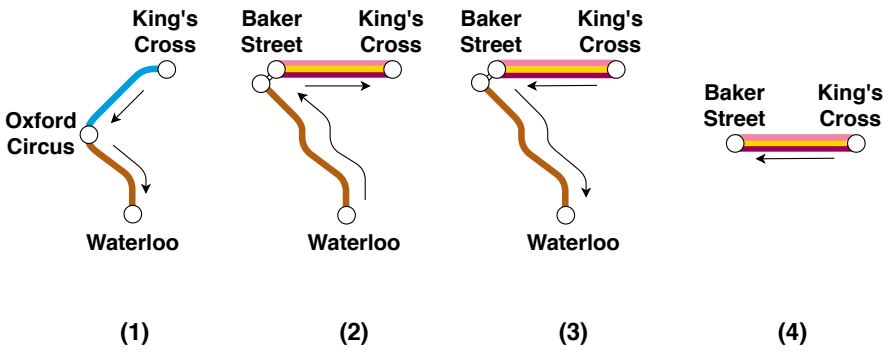


Figure 3.7: London Underground route types.

With respect to the layout of the source and destination platforms, we identify four categories of routes, which we also depict in Fig. 3.7:

1. Unique lines going through both source and destination platforms, e.g., King's Cross St. Pancras (light blue) - Oxford Circus - Waterloo (brown)

2. One line going through the source platform and multiple lines through

the destination platform, e.g., Waterloo (brown) - Baker Street - King's Cross St. Pancras (yellow/violet/pink)

3. Multiple lines going through the source platform and one line through the destination platform, e.g., King's Cross St. Pancras (yellow/violet/pink) - Baker Street - Waterloo (brown)

4. Multiple lines going through both source and destination platforms, e.g., King's Cross St. Pancras (yellow/violet/pink) - Baker Street (yellow/violet/pink)

We perform example experiments regarding the four different categories of routes, running 100 rounds and computing the mean accuracy for each. For comparison reasons, we fix the following settings: people that enter and exit a train (200), people following the analyzed route (100), people on the train to destination coming from other directions (200-100=100), total number of people on a platform having 3 lines going through it (500). Recalling that within (Λ, Γ)-crowd flows the values of λ and γ represent percentages, the routes can be mapped to crowd flows like this:

1. ({50},{50})-crowd flow
2. ({50},{200})-crowd flow
3. ({80},{20})-crowd flow
4. ({80},{80})-crowd flow

Table 3.1: London Underground routes accuracies.

Route type	**Accuracy (k=2)**	**Accuracy (k=3)**	**Accuracy (k=4)**
(1)	0.9502	0.94	0.9195
(2)	0.8742	0.8589	0.8493
(3)	0.8651	0.8443	0.8378
(4)	0.6194	0.5788	0.5774

The results of the experiments can be seen in Table 3.1. These results would be achieved using the already existing sensing infrastructure, without any modifications, as it is currently deployed in the London Underground Network. The impact that our anonymization process has on the accuracy of crowd monitoring queries concerning the first three types of routes is low. For the fourth type of route, we cannot accurately capture the crowd flow by using the existing sensing infrastructure alone. The reason is that we are trying to identify a relatively small crowd in relation to an overwhelming number of

leavers and joiners. A solution at hand for accurately capturing this situation would be augmenting the sensing infrastructure with scanners placed directly on the trains. Otherwise, measurements shall be done only for situations where either the source or the destination allows us to do accurate counting.

3.4 Related Work on Anonymity

Performing crowd-monitoring by leveraging the communication interfaces of the widely-available modern smartphones is currently done at large scale. Numerous ways of doing it are already out there, having different approaches on individuals' privacy or anonymization issues. Having already presented in Chapter 2 related work regarding pedestrian tracking, flow identification and privacy-preservation approaches, let us now focus on anonymity matters, k-anonymity and state of the art developments in this field influencing the work presented in this chapter.

3.4.1 Anonymity

Anonymity, as a means of achieving privacy, is defined in [79] as noncoordinatability of traits such that a person is nonidentifiable. Privacy regulations regarding data processing carefully consider this aspect. For example, GDPR recital 26 [38] states that if personal data is rendered anonymous in such a manner that the data subject is no longer identifiable, then data protection principles do not apply any more, thus indicating anonymization as a very powerful mechanism for achieving privacy. Along with this, it explicitly mentions pseudonymization as a counterexample. Our anonymization process is tailored in such a way that every individual present in *CMD* is proven to be protected under detection k-anonymity guarantees, no matter what crowd-monitoring query is performed.

First introduced in [66] by Samarati and Sweeney as a privacy-preserving policy for data releasing and then extended in [75], k-anonymity is defined as the kind of protection achieved when the information about a person contained in a release is indistinguishable from k-1 other persons. Opportunely, k-anonymity is indicated as an acceptable anonymization technique by the European Data Protection Board [19], making it a strong starting point when designing a mechanism to protect information about individuals under GDPR. In our case, the information to protect would be the mere presence of a person near a scanner during an epoch or in a crowd flow. This presence is indicated by the person having a unique identifier which is displayed among the results of a crowd-monitoring query. Hence, pursuing detection k-anonymity comes

naturally. However, to the best of our knowledge, there is no investigation performed regarding this particular setting.

Using k-anonymity alone is prone to several attacks, such as homogeneity attack and background knowledge attack, as suggested by Machanavajjhala et al. [52]. To avoid these, the authors propose another technique, l-diversity, to ensure that for every equivalence class of size greater or equal to k there are at least l well-represented values for the sensitive attribute. Considering the scanner and the epoch as nonsensitive attributes and the identifier as a sensitive attribute, we can clearly see that these attacks are not possible in our setting and l-diversity suddenly becomes a nonproblem. The reason is that detection k-anonymity is achieved by manipulating the original identifiers into anonymized identifiers occurring multiple times, so even if the anonymized identifiers in a query are all identical, in fact they correspond to distinct values. We do protect against another kind of diversity attack though, through l-surjectiveness, as previously described in this chapter.

Anonymization techniques based on k-anonymity are present in numerous domains; relevant to our work are approaches regarding location-based services (LBS), moving-objects databases (MOD), as well as trajectory databases. All have in common the spatio-temporal dimension of the data being protected. In [15], Bettini et al. look into k-anonymity for location-based services, where a geo-localized history of user requests to a service provider can reveal sensitive information about individuals. Indirectly, such history is, in fact, a trajectory, and can be related to persons being sensed in a crowd-monitoring environment. However, their solution, which is based on historical k-anonymity, does not work for our settings since it only ensures that there are at least k people launching requests across the same spatio-temporal history, thus not protecting an individual's presence per se. In [8], Abul et al. propose (k,δ)-anonymity for trajectories, such that there are at least k trajectories found within a cylinder of uncertainty having the radius δ. Trajectory translations should be performed until such conditions are met. This concept is very closely related to ours and, if we consider the scanners as central points and their ranges as δ, translations are not even required because our system does not store localization information other than the position of scanners. In other words, any trajectory would already be within such a cylinder. Even so, we do not have trajectories in our system, but detections that deem trajectories only after they have already reached *CMD*. This is why this solution cannot be applied on the fly, at the collection point, as we demand. For the same reasons, similar solutions presented in [57], [76] or [24] do not suit our problem.

Finally, there are several studies about ensuring k-anonymity on the fly for data streams, such as [84], [22] or [50]. In essence, these methods ensure

that streaming data is made k-anonymous before publishing, just like we do. In contrast to our work, all the existing works consider a setting in which a single trusted server collects and stores the *raw (nonanonymized)* stream of data (typically from one source) which it turns into a k-anonymous form before final publishing; this works by taking the complete history of the data stream into account for the anonymization procedure. Unfortunately, this approach does not work in our setting where the anonymization must happen at each data source in isolation (i.e. at each scanner in our case) before it reaches the server *and* without access to the history of the complete data stream that ultimately consists of the data from multiple sensors. This difference, i.e. the anonymization of data at each sensor in isolation as opposed to the anonymization at the central collection point, defines the major challenge that we tackle in our work.

3.5 Conclusion

In this chapter, we addressed the problem of privacy-preservation through anonymity in crowd-monitoring systems. Our aim was to ensure that the privacy of each monitored individual is preserved while the system can still offer meaningful insights regarding pedestrian dynamics. Having privacy-by-design principles in mind, we designed a lightweight anonymization process to be executed right on the crowd-monitoring sensors, before forwarding the data to a central server. This process manipulates the detected identifiers of individuals through a series of pseudonymization, truncation and correction operations. After these operations are executed, every individual whose smartphone is being monitored ends up being protected with anonymity guarantees, making our solution aligned with GDPR specifications. In our construction, we introduce detection k-anonymity as anonymity metric, ensuring that there is no crowd-monitoring query in which there are anonymized identifiers occurring fewer than k times each. Besides that, we introduce l-surjectiveness as a metric indicating the number of real devices behind any anonymized identifier.

We evaluated our anonymization process on pedestrian crowd flows, first in a purely simulated environment, then in an environment reproducing the real-life deployment from the London Underground Network. Results show that our anonymization process has a low impact on the accuracies of queries related to crowd flows suffering small perturbations, i.e. relatively few people leaving or joining the flow. In the realistic environment reproduction, the accuracy of such queries stays above 0.8 for all the tested values of k, topping at 0.9502 for k=2 in the case of a ({50},{50})-crowd flow. The impact is higher

for crowd flows suffering big perturbations and having a relatively small size in comparison with the number of leavers and joiners. However, this is a desirable result, as we designed our system to also protect the anonymity of people in small crowds, hence offering lower accuracy in those cases. Our experimental results confirm this behavior.

Chapter 4

Bloom Filters & Homomorphic Encryption

The aim of a crowd-monitoring system for pedestrian dynamics is to provide insights on crowds in the form of statistical counts. In the process of building such aggregated information, privacy-sensitive data of individuals has to be used. Following data protection as a goal and data minimization as a way to achieve it, we envision a system that offers statistical counts as the only accessible information in the clear while protecting the privacy-sensitive data of individuals at rest and during processing.

Searching for a method to facilitate oblivious crowd observance over time, in [70] we investigated the use of Bloom filters (BFs), i.e. probabilistic data structures supporting set operations, together with homomorphic encryption (HE), i.e. a type of encryption that allows performing operations on encrypted data. Preliminary experiments indicated that such a scheme can indeed be suitable for our goal, allowing counting over encrypted representations of sets or intersections of sets of devices without revealing what is being counted.

Having shown that by combining BFs with HE it is possible to provide statistical counts while protecting the data of individuals, in this chapter we address the problem of actually designing a crowd-monitoring system based on the proposed principles, as well as what has to be done in order to make the deployment of such a system feasible. The contributions of this chapter are summarized as follows.

- We propose a crowd-monitoring system that can produce statistical counts as a service for interested consumers while protecting the privacy-sensitive data of sensed individuals. The system is secure against honest-but-

curious adversaries and it allows counting crowds at one location, as well as counting the crowd flow between locations.

- We carry out an implementation of the system using Raspberry Pi as a typical sensing device and two different server configurations (i.e. a laptop and a more powerful cloud server) as operators under encryption, to assess the feasibility of our solution. We deploy the system, trialing different setup parameters, and analyze its performance when faced with a whole range of crowds.

- We perform a thorough evaluation using simulated data, to explore the potential of our solution to estimate statistical counts, as well as using real-world data from an open-air festival consisting of 26 million detections, to see how the system performs when dealing with real detections generated by actual pedestrian movements. Statistical counts regarding footfall are estimated with an accuracy of at least 97.2% when analyzing the most crowded area of the festival. 88.5% of the statistical counts on crowd flows happening during the festival on a circulated street have an accuracy of at least 90%, while 98.7% of the estimations are less than 3 devices away from the real counts.

This chapter is based on the work presented in [71]. The rest of the chapter is organized as follows. Section 4.1 introduces the system model, including crowd-monitoring formalities, involved actors and security requirements. Section 4.2 presents our construction instantiating the system model by combining BFs with HE. In Section 4.3 we perform an evaluation of the system on simulated data to get an understanding of how well statistical counts can be estimated. Section 4.4 presents an implementation of the system, together with a performance analysis. Section 4.5 evaluates our system on real-world data from a mass event, followed by a discussion in Section 4.6. Finally, Section 4.7 concludes the chapter.

4.1 System Model

Crowd-monitoring systems are usually deployed to provide an understanding of the pedestrian dynamics happening in crowded public spaces. Hence, the world of such a system is represented by a public space where crowds of people are expected. In this space, a technical infrastructure is usually installed to collect data about the crowd. Based on that sensed data, useful information is built in the form of crowd-monitoring insights. Considering that for building

these insights data related to people is handled by multiple entities, the whole process should be governed by some clearly stated rules to ensure that the privacy of the sensed individuals is protected. We start by first recapping the Wi-Fi crowd-monitoring environment. Then, we model the insights offered by the system as statistical counts for pedestrian dynamics. Eventually, we describe the actors involved in the process, together with requirements such that the privacy-sensitive data of individuals is protected.

4.1.1 Crowd-Monitoring Environment

A Wi-Fi crowd-monitoring system consists of a set of Wi-Fi scanners $\mathcal{S} = \{s_1, ..., s_n\}$ installed in a public space where crowds of people are expected. Devices carried by people emit probe requests containing the MAC address of the sender $a \in \mathcal{A}$ where $\mathcal{A} \subset \{0,1\}^{48}$. Each occurrence of such a device is associated with an epoch $e \in \mathcal{E}$. We call the 3-tuple (a, s, e) a *detection*, signifying that a device with MAC address a was detected by scanner s during epoch e. We model a crowd as a set of detections $\mathcal{D}_{s,e}$ containing the devices detected by a scanner s during an epoch e.

4.1.2 Statistical Counts for Pedestrian Dynamics

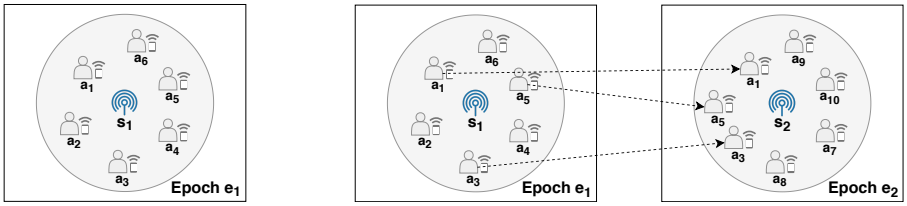
Detections made by Wi-Fi scanners can be used to derive numerous statistics on crowds. In particular, for pedestrian dynamics we are looking at two main situations in our work:

1. The crowd of people present in one place in a particular period of time, known as *footfall* (Fig. 4.1a)
2. The *crowd flow* of people traveling from one place to another (Fig. 4.1b)

We formally define these situations in terms of counts below.

Definition 11. Let $\mathcal{D}_{s,e}$ be the set of detections made by a scanner s during an epoch e within a crowd-monitoring system. We define the **footfall** in the area covered by the range of scanner s during epoch e as the count obtained by computing $|\mathcal{D}_{s,e}|$.

Definition 12. For a collection of sets $\{\mathcal{D}_{s_1,e_1}, \ldots, \mathcal{D}_{s_n,e_n}\}$ representing detections made by scanners $s_1, \ldots, s_n$ during epochs $e_1, \ldots, e_n$ within a crowd-monitoring system, we define the **crowd flow** as the intersection $\bigcap_{i=1}^{n} \mathcal{D}_{s_i,e_i}$ over that collection. The size of the crowd flow following the corresponding path is the count obtained by computing $|\bigcap_{i=1}^{n} \mathcal{D}_{s_i,e_i}|$.



(a) (b)

Figure 4.1: Situations encountered in pedestrian dynamics: (a) *footfall* and (b) *crowd flow*.

A Wi-Fi crowd-monitoring system concerned with pedestrian dynamics should use readings made by scanners to produce the necessary data such that these two types of statistical counts can be computed. Note, though, the differences between Definition 12 and Definitions 3 and 4 presented in Chapter 3. Those two define the crowd flow in relation with the percentages of joiners and leavers, while the current definition is independent of such parameters.

4.1.3 Service Model

Let us now model, from an architectural point of view, the actors taking part in the process. We separate data gathering and processing from its usage and propose two classes of entities (see Fig. 4.2).

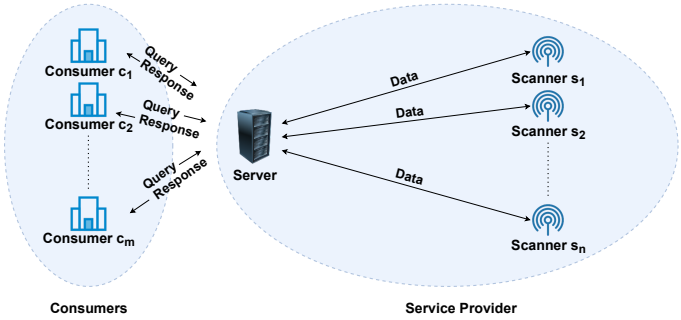


Figure 4.2: Service Model.

Service Provider (SP). This is the entity which owns the sensing infrastructure and provides the crowd-monitoring service. The service is provided in

the form of responses to queries received from parties interested in pedestrian dynamics insights. The responses should contain sufficient information to allow the computation of statistical counts corresponding to the situations indicated by the queries. Queries can be numerous and they can span across multiple scanners and epochs, so we assume that a separate service, acting as a central manager, assembles the data generated by scanners into responses. This service can be implemented, e.g., by a single or multiple cloud-based servers. For coherence, we are going to use the term *server* throughout the rest of the chapter.

Consumers. These are public or private parties interested in understanding pedestrian dynamics. They are external to the crowd-monitoring system, launching queries whenever they want to discover insights. Also, they have to process received responses in order to find out the desired statistical counts.

4.1.4 Security Requirements

Privacy-sensitive data related to individuals is handled throughout the crowd-monitoring process. This includes unique identifiers, places and times of detections. Improper handling can lead to infringing the privacy of individuals. Therefore, we impose a number of security requirements that must be satisfied while still being able to compute statistical counts.

Consumers. The only information we allow consumers to learn is that of statistical counts as answers to queries they launch, namely the count of all detections made in either footfall or crowd flow situations. Additionally, consumers need to have a publicly verifiable identity (e.g., a public key certified by a trusted certificate authority).

Scanners. We demand that scanners are tamper-proof, such that the system can put trust in the outputs they produce. Nevertheless, even if their outputs can be trusted, we do not allow scanners to be in possession of data other than what they can generate themselves through sensing. We do allow them, though, to be aware of the consumers enrolled in the system, as they might need to generate specific data for each consumer. Still, scanners should be programmed to accept only certified consumers. Finally, at the end of each epoch scanners must discard all the detections made in that epoch, thus keeping data in clear as short as possible.

Oblivious server. The server should not be able to assemble any meaningful information from the data it handles, neither MAC addresses collected by scanners nor statistical counts nor other intermediary information related to

individuals. Fulfilling this requirement protects individuals from honest-but-curious SPs, as well as in the case of an external attack on the server. Following the same honest-but-curious model, despite the server trying to extract as much information as possible from the data it handles, yet we trust it to run the protocol correctly. So we allow it to know of involved consumers, scanners and queries, as it needs to manage incoming queries as well as obliviously assembling their responses.

Non-colluding entities. We do not allow the SP to collude with any of the enrolled consumers. This means that the SP and consumers cannot cooperate outside the protocol to derive information in addition to what they are allowed to know according to the protocol. It also means that the SP cannot enrol itself as a consumer (situation prevented, nevertheless, by a previous requirement as consumers must have publicly verifiable identities).

4.2 Our Construction

As an instantiation of the previously introduced system model, we present our construction supporting statistical counts for pedestrian dynamics as a service while fulfilling the proposed security requirements.

Each epoch, scanners collect detections and write them in detection sets. In the case of a footfall query, a scanner can simply calculate itself the cardinality of such a detection set, delivering it as a response to the server which forwards it to the intended consumer. Then, the scanner can immediately discard the data used for the calculation, as demanded by our design choice. The problem gets complicated when queries regarding crowd flows are launched. The answer to a crowd flow query is represented by the cardinality of an intersection of sets coming from multiple scanners and epochs. As detection sets are discarded, by design, at the end of each epoch, we are faced with the challenge of performing an intersection of sets without having the original sets any longer. In consequence, to support this we should come up with structures resembling sets and allowing intersections, but without knowing what is stored in the structures themselves.

4.2.1 Bloom Filters

A Bloom filter (BF) [18] is a space-efficient probabilistic data structure typically used for storing sets of elements and allowing set membership testing. It consists of an array of m bits initially set to 0, along with k different hash functions. Whenever a set element e has to be added in the BF, the k hash functions

are computed on e, each result pointing to one of the m array positions; these positions are set to 1. Correspondingly, to check whether an element is a member of a set, one has to verify if all the positions indicated by the k hash functions are set to 1. From this it immediately follows that by multiplying the bits found on the same positions of BFs which were previously encoded using the same hash functions, a new BF is obtained which approximately corresponds to the intersection of the underlying sets.

False negatives cannot occur with BFs, but false positives can, since positions corresponding to an element can be set to 1 by hashes of elements other that the one being tested for. Nevertheless, the parameters of the BF can be tuned in such a way that a desired probability of false positives p is achieved when knowing that a set of n elements must be accommodated [21].

We consider to use BFs in our system to support the result computation for footfall and crowd flow queries as the cardinality of the underlying sets. According to Swamidass and Baldi [74], the cardinality c of a BF having t bits set to 1 can be estimated as:

$$c = -\frac{m}{k} \ln(1 - \frac{t}{m}) \tag{4.1}$$

Service provision could thus happen in the following way. Scanners encode detections they make into BFs and transfer them, at the end of each epoch, to the server. When the server receives a query from a consumer, it starts assembling an answer by gathering the necessary data generated by scanners. In case of a footfall query, it uses that single BF of interest, while for a crowd flow query it generates a new BF by performing a bitwise multiplication of the corresponding BFs. Afterwards it shuffles the positions to remove any meaning, transforming the BF into a random array of 0's and 1's, and delivers the result to the consumer. We can trust the server to do the shuffling as this is part of the protocol and it is assumed to correctly follow it. The consumer, being provided with the values of m and k, computes the desired statistical count by applying eq. (4.1), as the number of 1's is not affected by the shuffling.

Until now we have a system that can address both footfall and crowd flow queries as specified by the service model. At the same time, security requirements are met at scanner and consumer level. However, the solution is not complete, the server not being yet compliant with our requirements. The MAC address space is easily enumerable [16]. Because of this, an entity (including the server) knowing the hash functions can do an exhaustive search on a BF in limited time, revealing with high probability the MAC addresses stored in it. Moreover, the server can apply itself eq. (4.1) and find out any

statistical counts it desires, since it has access to BFs in the clear and it sees how many 1's are in each of them.

To prevent the server from doing the previously mentioned actions, we need to combine BFs with an encryption scheme such that the server would handle only encrypted data that it cannot decrypt. To satisfy our security requirements while maintaining the functionality unchanged, such an encryption scheme should have the following properties:

- Encrypting the same value on several occasions should produce different ciphertexts. Otherwise, as BFs contain only 0's and 1's, an attacker could learn, despite encryption, the positions containing identical values, thus being able to infer how the original BFs looked like.

- It should allow multiplications under encryption, such that the decryption of the result of a multiplication of ciphertexts equals the result of the multiplication of the unencrypted values behind those ciphertexts.

- It should be a public-key encryption scheme such that everyone (especially the scanners) can encrypt values but that only the consumer, who is the only one having the corresponding secret key, can decrypt it.

4.2.2 Homomorphic Encryption

Homomorphic encryption (HE) [64] is a type of encryption that allows mathematical operations to be performed directly on encrypted data without requiring decryption. The results of such operations, encrypted as well, are the same as if the operations were performed on the unencrypted data. HE includes different classes of encryption schemes depending on how many types of operations they allow and how many times they can be applied. Partially homomorphic encryption (PHE) is a class of schemes that allow performing a single type of operation under encryption, either addition or multiplication, for an unlimited number of times. PHE sufficiently satisfies our requirements, having specified in 4.2.1 that we are looking for an encryption scheme that allows multiplications under encryption (i.e. a single type of operation).

ElGamal [37] is such a PHE cryptosystem which allows multiplications under encryption; we are going to use it in our construction. The algorithm is asymmetric, using a public key for encryption and a private key for decryption. Furthermore, the algorithm is probabilistic, involving randomness in the encryption process, so that encrypting the same value several times yields different and indistinguishable ciphertexts.

Combining BFs with HE closes the circle of our system model. The complete service provision happens as follows. When a consumer enrolls in the system, it generates a public-private key pair and it gives the public key to the SP, which distributes it to its scanners. At the end of an epoch, scanners write detections into a BF. For each enrolled consumer, they make a copy of that BF and encrypt each position with the public key of that specific consumer.[1] They then send the resulting encrypted BFs (EBFs) to the server and discard the original detections. When the server receives a query from a consumer, it acts in the same way as before, just that this time it obliviously handles encrypted data and, if necessary, it performs bitwise multiplications under encryption, as depicted in Fig.4.3. To estimate the statistical count from a response, a consumer iterates through it, decrypts the ciphertexts, sums up the 1's to find out t and applies eq. (4.1).

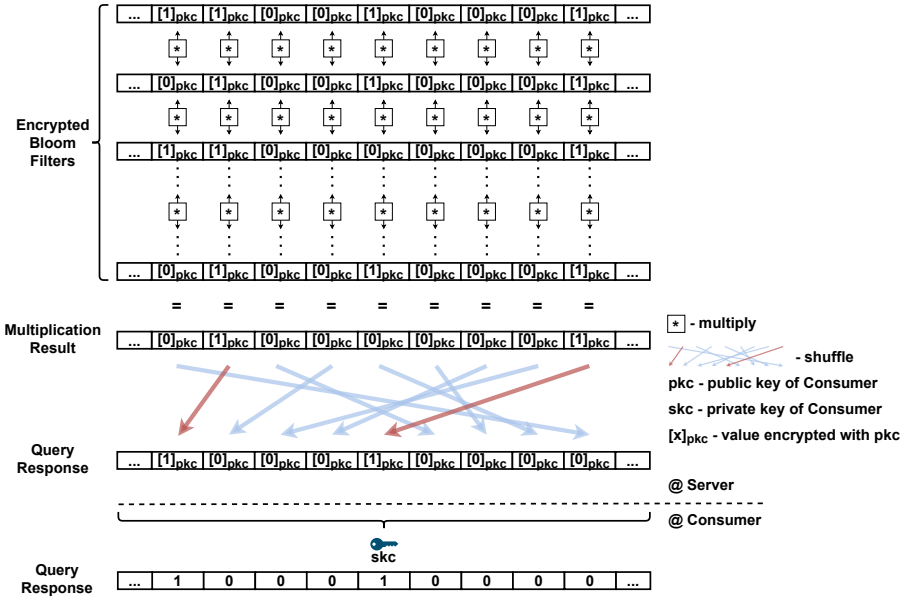


Figure 4.3: Preparing response to a crowd flow query by performing multiplications under encryption and shuffling. Consumer decrypts response, then counts the 1's and estimates the statistical count using eq. (4.1).

[1] We note that 0's are represented as random numbers as ElGamal can deal only with positive integers.

4.3 Accuracy Analysis

In this section we explore the potential of our solution to estimate statistical counts for pedestrian dynamics. We generate detections emulating footfall and crowd flow situations for a whole range of sizes. We feed these detections as inputs to our system and perform an accuracy analysis of the responses, i.e. comparing the statistical counts generated through our system with the actual counts of the emulated situations. In doing this, we examine different combinations of parameters for BFs, as this is the part in our solution that influences the statistical counts. For the hashing part, we choose to use MurmurHash3 [12] with different seeds, a fast hash function which despite its non-cryptographic nature is suitable for our case where the positions written in BFs are not available in the clear.

We expect to see differences between what is estimated and ground-truth values because of two main reasons. First of all, the result of the estimation formula is the number of elements having the *highest likelihood of being members*, given the state of the BF; the actual number can be different depending on the probabilistic properties of the underlying set. Secondly, *false positives* can be encountered when working with BFs, which means that the k positions corresponding to an element tested as present could have been set to 1 by the hash values of other elements; this can also lead to differences in counts when intersecting BFs for crowd flows.

The accuracy metric that we propose measures the closeness of the estimated count c to the real count c_t and is formally represented below. Please note that we choose to interpret estimated counts that are more than twice as high as the real counts as having accuracy 0 since they are more than 100% off from the real counts.

$$Acc = max(1 - \frac{|c - c_t|}{c_t}, 0) \tag{4.2}$$

BF parameters can be set in such a way that a desired false positive probability p is obtained when having a set containing n elements. The length m of the BF can be computed as $-n \ln p/(\ln 2)^2$ and the optimal number of hash functions k as $-\log_2 p$. In our crowd-monitoring setting, choosing values for n and p implies accommodating a maximum number of devices n detected by a scanner during an epoch while allowing a maximum false positive probability p for the resulting BFs. In Table 4.1 we display the setup parameters that we use throughout the rest of the chapter.

Table 4.1: BF Parameters.

p ↓ n →	100	1000	10000	100000	k ↓
0.0001	m=1918	m=19171	m=191702	m=1917012	13
0.001	m=1438	m=14378	m=143776	m=1437759	10
0.01	m=959	m=9586	m=95851	m=958506	7
0.1	m=480	m=4793	m=47926	m=479253	3

4.3.1 Accuracy of Footfall Queries

In principle, a lower m would be desirable for performance reasons, as it would mean less cryptographic operations to be performed. However, reaching a lower m requires opting for a higher probability of false positives p, which may have consequences on the accuracy of queries. To get a clear understanding of the link between the choice of p and the accuracy of footfall queries, for fixed values of n, we run experiments ranging p, as displayed in Table 4.1. As identifiers, we generate random MAC addresses coming from a uniform distribution, resembling, thus, real-world deployments, where a cryptographic hash is usually applied on the real identifiers before being processed; we will use the same way of generating addresses throughout the rest of this section.

We plot in Fig. 4.4, as worst case, the minimum mean accuracy measured for each choice of n and p when handling between 0 and n devices, with a step equal to 10% of n and doing 100 runs per step with different addresses each run. Results show that indeed lower accuracy is to be expected for footfall queries when choosing a higher p, a trend which is consistent across all the tested values of n. However, the impact is not significant, as even for the highest tested value of p, the accuracy does not get below 96.7%, 98.9%, 99.6% and, respectively, 99.8% for the 4 different values of n. For example, a worst-case accuracy of 98.9%, as when n is 1000 and p is 0.1, comes from estimating 784 instead of 800 devices.

By looking at the formula for estimating statistical counts (eq. (4.1)), when BF parameters are fixed, we see that the estimation and hence the accuracy of the concerned footfall query depend only on the number of bits t set to 1. In our case, t is dictated by the number of sensed devices c_t. Let us see, thus, what accuracy we can expect from footfall queries when *ranging* c_t. To see the trend, we run an experiment in which we fix n to 1000 and p to 0.01 and range c_t, starting from 0, going to n and then even beyond, up until it leads to a t getting close to m. We increase c_t with a step of 100, we do 100 runs with different addresses for each step and plot mean accuracies together with standard deviations in Fig. 4.5.

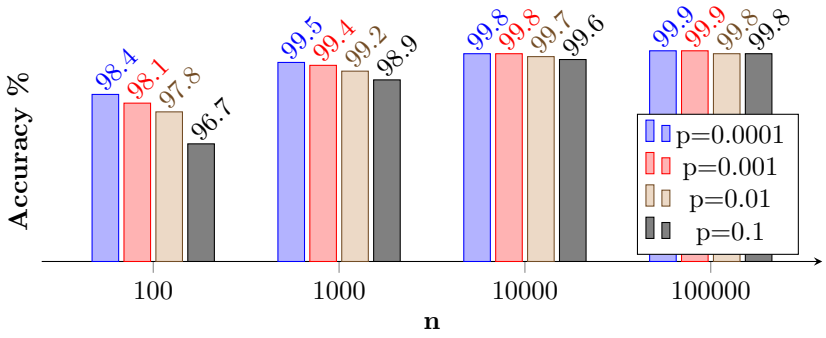


Figure 4.4: Worst-case accuracy for footfall queries when dealing with different values of n and p.

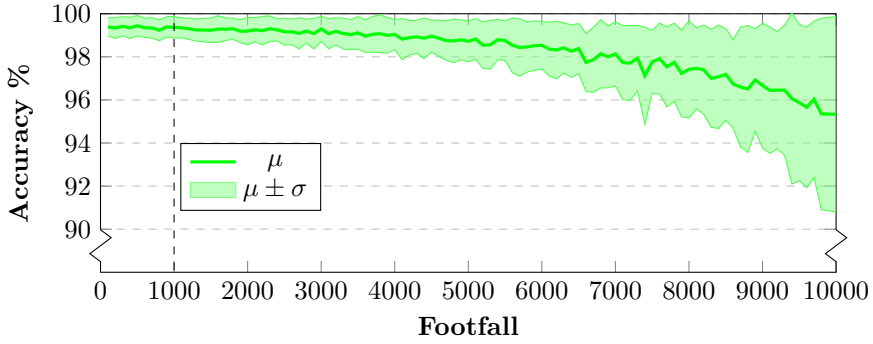


Figure 4.5: Accuracy of footfall queries when increasing the number of detected devices until completely filling up the BF. Parameters: n=1000 and p=0.01. The vertical dashed line marks n.

The accuracy of the statistical counts stays above 99.2% when $c_t \leq n$, as also presented in Fig. 4.4, showing a standard deviation of 0.04% in that worst case. For this experiment, the threshold for which c_t leads to a full BF is around 10000 devices. Regarding footfall only, the accuracy of the statistical counts stays above 95.3% even when larger-than-designed-for crowds of up to the mentioned threshold arrive, though with a higher standard deviation of 4.5%. We have run experiments with other values of n and p as well and we confirm that the same trend, with accuracies in the range of 90%'s, is to be seen. Please note, however, that in cases when the crowd is inflated beyond

the designed maximum, the false positive probability inflates as well, which deems the results as problematic for further counting crowd flows.

4.3.2 Accuracy of Crowd Flow Queries

A crowd flow is represented in data as a BF resulting from performing a bitwise multiplication of other BFs. This result is an approximation of the intersection of the underlying sets, as the probability of false positives leads to having false matches in an intersection. We say that the BF resulting from the bitwise multiplication of BFs which represent sets of detections is different from the BF representing the intersection of those sets of detections. It tends to be more different when more false matches occur. For this reason, in the same manner, the statistical counts estimated by eq. (4.1) will also get farther from the actual counts. This aspect was also noticed by Papapetrou et al. in [59]. They propose to improve the estimation by taking into account, besides the 1's in the resulting BF, the 1's in the BFs used in the bitwise multiplication. For t_1, t_2 and $t_\wedge$ as the number of bits set to 1 in two BFs to multiply and, respectively, in the result, the estimation formula is:

$$c_\wedge = \frac{\ln(m - \frac{t_\wedge \times m - t_1 \times t_2}{m - t_1 - t_2 + t_\wedge}) - \ln(m)}{k \times \ln(1 - \frac{1}{m})} \tag{4.3}$$

We run a preliminary experiment in which we fix n to 1000 and p to 0.01. We intersect two crowds of 500 people each and range the crowd flow size between 10 and 500 people. We plot the accuracy of statistical counts while using both estimation equations to understand the dimension of the improvement. In Fig. 4.6 we see that the improvement is sensible, the accuracy much faster approaching 100% when using eq. (4.3) instead of eq. (4.1).

For applying eq. (4.3), a consumer would need to know, along the crowd flow query response, the answers to the associated footfall queries in order to determine t_1 and t_2. This is not in contradiction with our system model, as consumers are allowed to launch such queries. Also, it does not imply any additional computationally expensive operations. Hence, we use this alternative in the rest of the evaluation for estimating statistical counts on crowd flows.

As we have previously mentioned, the probability of false positives p is an important parameter when it comes to estimating statistical counts on crowd flows. Inserting elements in a BF with higher p leads to a higher density of 1's than by inserting the same elements in another BF with lower p; consequently, this leads to more false matches when using the former type for estimating

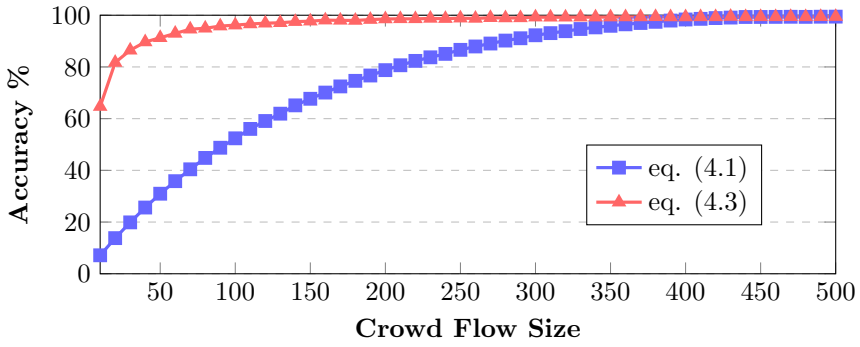
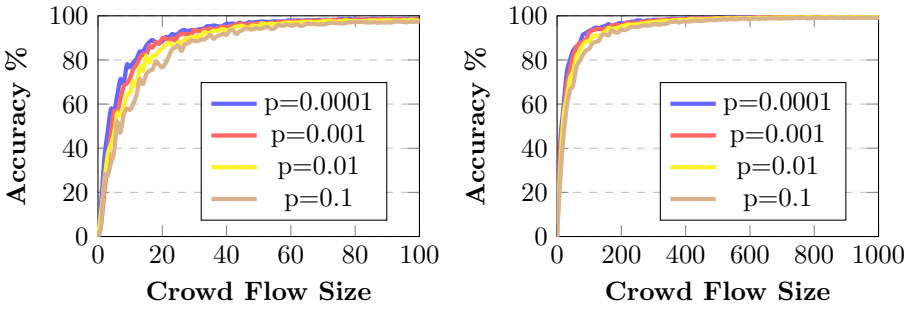


Figure 4.6: Preliminary experiment with n=1000, p=0.01, crowd size 500 in both locations, crowd flow size ranges between 10 and 500.

crowd flows. To assess how big of a problem this is, we propose the following experiment. For a fixed n, resembling a worst-case scenario, we take two crowds of size n (i.e. the maximum accommodated number of devices) and range the size of the crowd flow happening between them from 1% to 100% of n. We plot the mean accuracy (from 100 runs) of the estimated crowd flows when choosing p to be 0.0001, 0.001, 0.01 and 0.1. We run the experiment four times, for n equals 100, 1000, 10000 and 100000. We display the results for the first two values of n in Fig. 4.7.



(a) n=100 (b) n=1000

Figure 4.7: Accuracy of crowd flow queries for n first fixed to 100 and then to 1000, for different values of p, when ranging the crowd flow size between 0 and n. Plots display worst-case scenario, crowds at the ends of the crowd flow being fixed to the maximum value (i.e. n).

The accuracy of statistical counts on crowd flows shows logarithmic growth for all the setups we have tested, from low when the crowd flow is small in comparison with the intersected crowds to high for more steady crowd flows. Also, as expected, higher p means constantly having lower accuracy, but not significantly lower; e.g., for n equals 10000 and 100000, lines representing the accuracy for different p-s almost entirely overlap, this being the reason for not showing their graphs. This is important, as we recall that p inversely influences the length m of BFs (see Table 4.1) and, thus, the performance of the system, a phenomenon which we will study in detail in the upcoming section. Another pattern we observe is that for setups where the maximum crowd size is higher, the accuracy gets close to 1 quicker. To illustrate this better, we draw another graph (Fig. 4.8) based on the same experiment, this time showing, for a fixed value of p and all the four values of n, the minimum crowd flow size as a percentage of n for which accuracies of at least 50%, 60%, 70%, 80% and 90% are reached. To reach, for example, an accuracy of at least 90%, the crowd flow size should be at least 29% of the initial crowds when n is 100, 10.8% when n is 1000, 3.7% when n is 10000 and 1.3% when n is 100000. We remind, though, that these are results for worst-case scenarios, when initial crowds are equal to n; when initial crowds are smaller, accuracy thresholds are reached even quicker.

As seen from Fig. 4.7, a low accuracy is to be expected for crowd flows that are small in comparison with the intersected crowds, but a low accuracy does not necessarily mean something bad for crowd-monitoring purposes, especially when talking about small numbers. For example, having two crowds of 500 people with an actual crowd flow of 20 people between them and an estimated count of 15 or 25 incurs an accuracy of 75%, which might seem low at first sight. In reality, the estimation is only 5 devices away from the real count, which can be very well considered an insignificant error given the size of the initial crowds, i.e. being 100 times bigger. This leads us to looking into the actual distances between the estimations and the real counts, to get better insights for such situations where the accuracy does not tell too much. For this, we run an experiment in which we fix p to 0.01, n to 1000, initial crowds to n and range the size of the crowd flow between 0 and 100 with a step of 20. We do 1000 runs per step, each time using different addresses for devices. We show the results in Fig. 4.9. For the upper part of the graph, we compute the mean estimated count μ_c, as well as the standard deviation σ_c. We plot the difference between μ_c and the real count c_t, along with σ_c, to see how far away from the real counts, corresponding to 0 on the y axis, and in which direction the estimations are. In the lower part of the graph we plot σ_c as a percentage of μ_c, a measure commonly known as *relative standard deviation*.

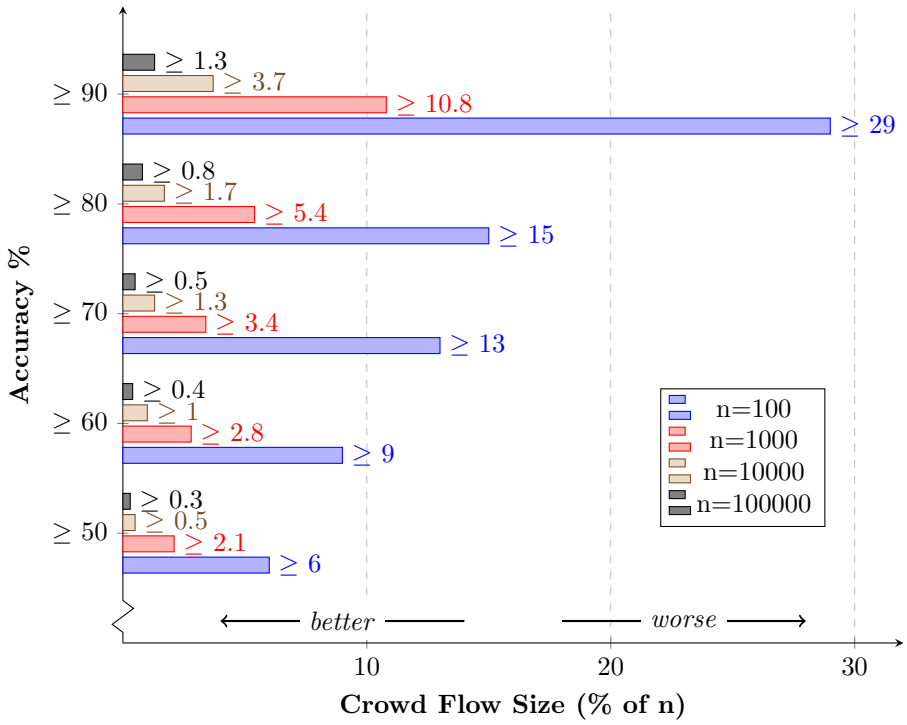


Figure 4.8: Fixing p to 0.01 and initial crowds as worst-case to maximum (i.e. n), we display the crowd flow size as a percentage of n for which accuracies of at least 50%, 60%, 70%, 80% and 90% are reached.

The maximum standard deviation is 14.32 when the real count is 40 devices, a point beyond which the mean estimation stabilizes as almost identical to the real count. It decreases to the right, the estimations getting closer to the mean. It also decreases to the left in a counterintuitive manner, due to a positive estimator bias inflicted by estimations which are negative, according to the formula, but we set them to 0, as statistical counts cannot be negative. The relative standard deviation quickly decreases, as expected from previous experiments concerning accuracy, the estimations getting closer to the real counts as the crowd flow size increases. Considering the case of the maximum standard deviation, 68% of the estimations of a crowd flow of 40 devices traveling between two crowds of 1000 devices fall between 26.63 and 55.27, with a mean of 40.95 devices. The minimum encountered standard deviation is

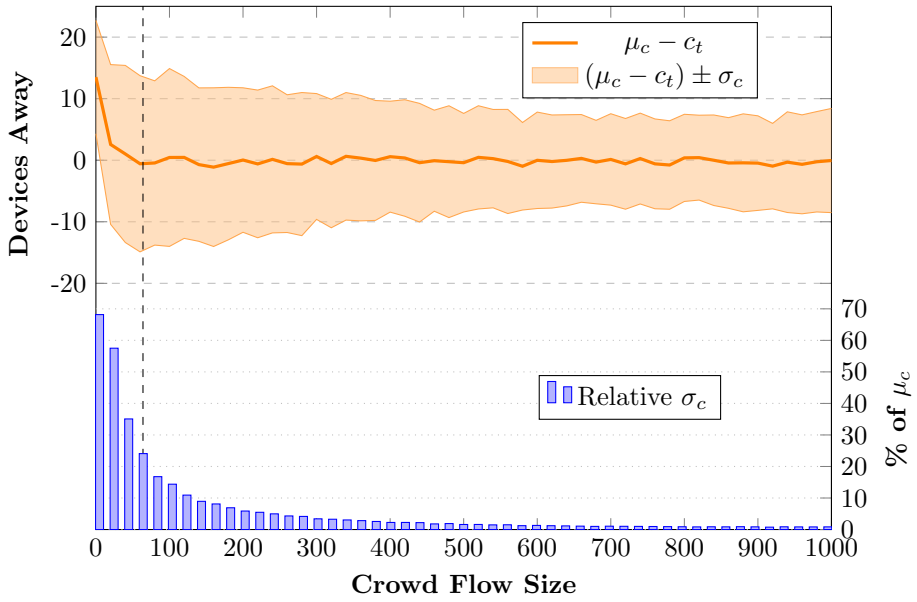


Figure 4.9: We fix p to 0.01, n to 1000, initial crowds as worst-case to maximum (i.e. 1000) and range crowd flow size between 0 and 1000. In the upper part we display mean estimated counts as *devices away* from real counts, together with standard deviations of the estimated counts. Below we plot standard deviations as percentages of the mean estimated counts. The vertical dashed line marks the minimum crowd flow size for which estimations always turn positive.

6.78 for a real count of 720 devices; for this case, 68% of the estimations fall between 714.21 and 727.77, with a mean of 720.99.

4.4 Implementation & Performance Analysis

To understand the cost dimensions our solution incurs at different levels of the system and bearing in mind that adding homomorphic encryption may generate considerable overhead, in this section we carry out an actual end-to-end implementation. We describe the necessary algorithms and we deploy them at scanner, server and consumer. Based on this implementation, we conduct a performance analysis addressing several relevant and practical concerns. We use again MurmurHash3 with different seeds as a hash function for BFs. We

instantiate ElGamal using the NIST P-256 elliptic curve [9] and we use the SCAPI[2] library [35] for homomorphic encryption support.

4.4.1 Scanner-Side

Scanners execute Algorithm 2, which takes each detection made in an epoch, applies the k hash functions on it, sets the corresponding BF positions to 1 and then it applies the ElGamal encryption. They run this algorithm for each consumer enrolled in the system.

```
Input: pkc //Public key of consumer;
Input: DSE[ ] //Detections;
Input: m //BF length;
Input: k //Number of hash functions;
Input: H[k] //The hash functions;
Output: EBF[ ] //Encrypted BF;
BF := [0];
/* Set BF positions corresponding to detections */
foreach DSE as currentMAC do
    for i := 0 to k − 1 do
        pos := H[i](currentMAC);
        BF[pos] := 1;
    end
end
/* Encrypt each BF position */
for i := 0 to m − 1 do
    if BF[i] = 1 then
        EBF[i] := ElGamalEnc(pkc,BF[i]);
    else
        EBF[i] := ElGamalEnc(pkc,rand());
    end
end
return EBF;
```

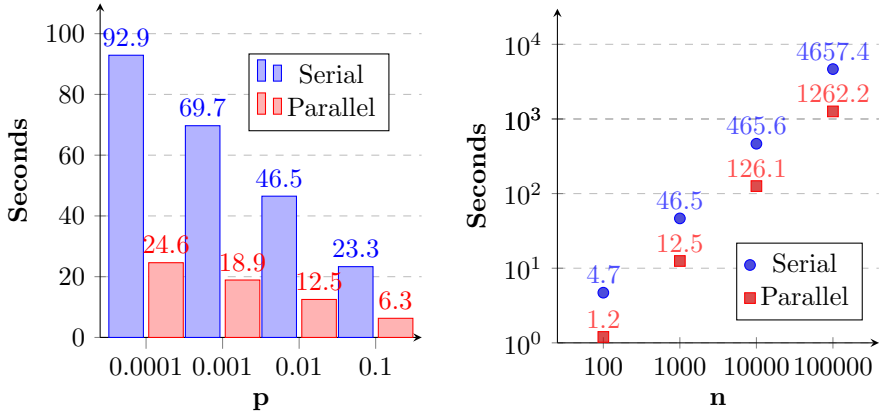
Algorithm 2: Algorithm running on a scanner, for each enrolled consumer, at the end of an epoch.

We implement this on a Raspberry PI 4B, which uses a Broadcom BCM2711 SoC, with a 1.5GHz 64-bit quad-core ARM v8 Cortex-A72 processor, 8GB of

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

DDR4 RAM memory, 16GB microSD memory card and it is running Ubuntu 20.10 as OS. We implement both serial and parallel versions of the algorithm, using C++11 threads for parallelization.

Supposing that a single consumer is enrolled in the system, we want to see how long it takes for a scanner to process the readings in an epoch. Analyzing the algorithm, for $d = |DSE|$, we see that for an epoch a scanner must compute $d * k$ hashes and m ElGamal encryptions. In Fig. 4.10 we display the results of an experiment timing such executions for serial and parallel implementations. In the former subfigure, we show the timings when p influences the run time, n being fixed to 1000 (axis y linear), while in the latter we fix p to 0.01 and range n (axis y logarithmic). Hash operations, in comparison with encryptions, are negligible, their duration falling in the range of nanoseconds, while encryptions are in the milliseconds range. This is why the values in Fig. 4.10 are mostly dictated by the resulting m for given p and n. For a crowd of maximum 1000 people, readings can be processed by the chosen scanner in less than 93 seconds for any tested value of p in the serial implementation and in less than 25 seconds for the parallel implementation; when p=0.1, the parallel run time is as low as 6 seconds. Larger crowds can be processed too in a reasonable amount of time judging by the parallel timings, such as approximately 2 minutes for a crowd of 10000 and 20 minutes for a crowd of 100000, the maximum size we have tested for.



(a) n fixed to 1000

(b) p fixed to 0.01

Figure 4.10: Time needed for processing the readings in an epoch for a single consumer.

To keep up with the incoming detections, a scanner must process the readings for all the consumers enrolled in the system in a time period which is less than the epoch length. Therefore, for a scanner achieving an average hashing duration in parallel t_h, an average encryption duration in parallel t_e, for the system parameter epoch length e_l and BF parameters k, n and m, the maximum number of consumers enrolled in the system at the same time can be computed as:

$$n_c = \lfloor \frac{e_l - k * n * t_h}{m * t_e} \rfloor \tag{4.4}$$

In particular, for the resource-constrained device which we used as a scanner and a fixed epoch length of 5 minutes, we plot n_c in Fig. 4.11 when ranging n for different values of p. More consumers can be supported for higher values of p. For example, when n is 10000, n_c is 1, 1, 2 and 4 for p equals 0.0001, 0.001, 0.01 and 0.1. The x axis extends up until no more consumers can be supported for the lowest value of p, which is when n reaches 11614. For the highest value of p, at least one consumer can be accommodated up until n reaches 46455, while for crowds of 1000 people, no less than 46 consumers can be satisfied.

4.4.2 Server-Side

A server gathers and stores EBFs from its scanners. Whenever it receives a query from one of the enrolled consumers, it runs Algorithm 3. Essentially, it multiplies under encryption, position-wise, the EBFs corresponding to the scanners and epochs concerned by the query and then shuffles the results.

For a query of complexity q representing the number of EBFs to combine, a server must do $m * q$ homomorphic multiplications under encryption, plus a final shuffle operation. We plot in Fig. 4.12a the computational effort needed to address queries of complexities between 1 and 10 for different values of p, where we consider a homomorphic multiplication to take one computational unit and the shuffle operation as negligible.

We implement Algorithm 3 both on a laptop and a more powerful cloud server to see how quickly different servers can provide responses to queries launched by consumers. We use the same ElGamal configuration as for the scanners. The algorithm is embarrassingly parallel, the iterations through the i loop showing no interdependencies, so we parallelize it using C++ 11 threads. The configuration of the laptop is running Ubuntu 20.04 x86_64, with 8GB RAM and a 4-core Intel(R) Core(TM) i5-10210 CPU @ 1.60GHz, while the cloud server is running Ubuntu 18.04 x86_64, it has 16GB RAM and a 16-core

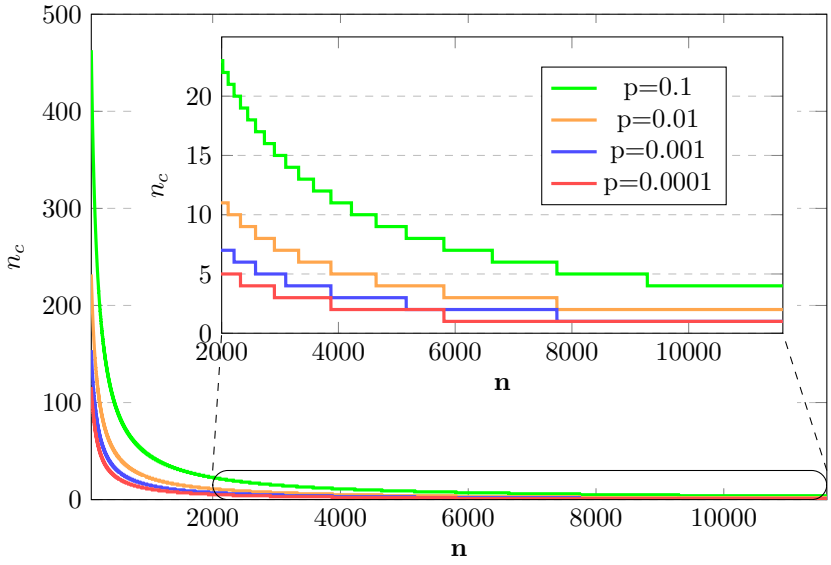


Figure 4.11: The number of consumers n_c enrolled in the system which can be supported by a scanner, when the epoch length is fixed to 5 minutes.

Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz. In Fig. 4.12b we plot the mean query response time in seconds from 10 runs on the two server configurations, as well as minimum and maximum response times, when q ranges between 1 and 10 and BF parameters are fixed. The responses are almost instant for footfall queries, only shuffling being necessary. The response time increases with q as expected from Fig. 4.12a, remaining below 10 seconds for the most complex query launched on the cloud server configuration, while even the basic laptop could provide responses in comparable times. Most importantly, almost ideal speedup can be easily achieved in a cloud environment by adding additional cores next to the existing ones, as the most computationally expensive parts of the algorithm are executed in parallel.

Besides response time, we also evaluate what throughput can be achieved by the two server configurations. We first show in Fig. 4.13a the throughput in *homomorphic multiplications under encryption per minute* (HMs/min), which is independent of BF-related parameters and independent of the complexity of launched queries. Having this information, a SP can form an idea on how to choose the parameters of the system in harmony with the expected crowd sizes, the accuracy desired from the system, the number and complexity of

```
Input: pkc //Public key of consumer;
Input: EBFs[ ][ ] //Encrypted BFs to combine;
Input: q //EBFs length;
Input: m //BF length;
Output: QRes[ ] //Query response for consumer;
EBFRes := [ ];
for i := 0 to m − 1 do
    mul := EBFs[0][i];
    for j := 1 to q − 1 do
        currentEBF[] = EBFs[j];
        mul := ElGamalMul(mul,currentEBF[i]);
    end
    EBFRes[i] := mul;
end
/* Rearrange positions in random order */
QRes := shuffle(EBFres);
return QRes;
```

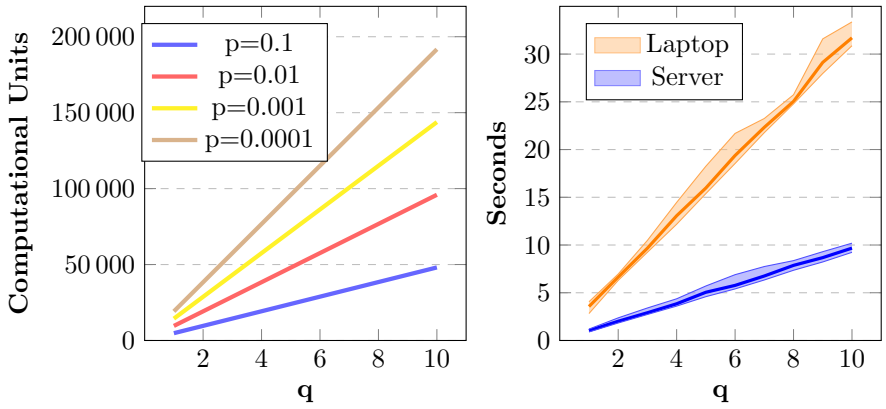
Algorithm 3: Server assembling query response for a consumer.

the queries expected from consumers. To exemplify, in Fig. 4.13b we then evaluate the throughput in *queries per minute* achievable by a system which must accommodate crowds of up to 1000 people, with a probability of false positives 0.01. We display on the graph only results for query complexities q between 2 and 10, as for footfall queries (i.e. $q = 1$) no additional homomorphic operations have to be done. In other words, for such setup parameters, a server can address approximately 35000 footfall queries per minute or tens of crowd flow queries per minute, with numbers decreasing when queries' complexity rises.

4.4.3 Consumer-Side

The statistical count related to a query is estimated by the consumer based on the response received from the server in Algorithm 4.

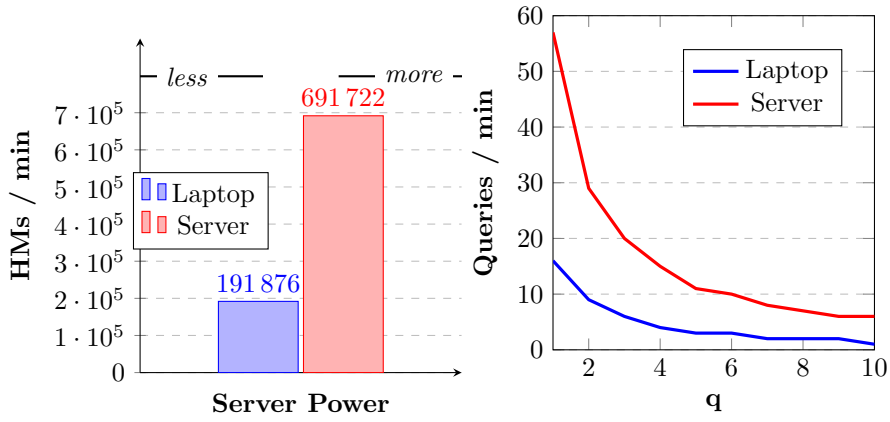
The computational load is approximately equal to that of a server assembling a response for a query of complexity 1, as the ElGamal multiplication and decryption operations have similar duration and the rest of the operations can be considered negligible. We plot in Fig. 4.14 the time needed by the laptop configuration acting as a consumer, when ranging m between its minimum and



(a) n=1000, showing computational units for different values of p

(b) n=1000, p=0.01, showing mean, min and max runtime in seconds for laptop and server configurations

Figure 4.12: Computing the answer for a query on a server for query complexities between 1 and 10.



(a) Throughput in HMs per minute

(b) Throughput in queries per minute when ranging q

Figure 4.13: Throughput for laptop and server configurations.

maximum values in Table 4.1. It ranges between less than a second for the

```
Input: skc //Private key of consumer;
Input: m //BF length;
Input: k //Number of hash functions;
Input: QRes[ ] //Query response;
Output: c //Estimated statistical count;
t := 0;
for i := 0 to m − 1 do
    x := ElGamalDec(skc,QRes[i]);
    if x = 1 then
        t := t+1;
    end
end
/* Estimate count */
c := -m/k*ln(1-t/m);
return c;
```

Algorithm 4: Statistical counts estimation.

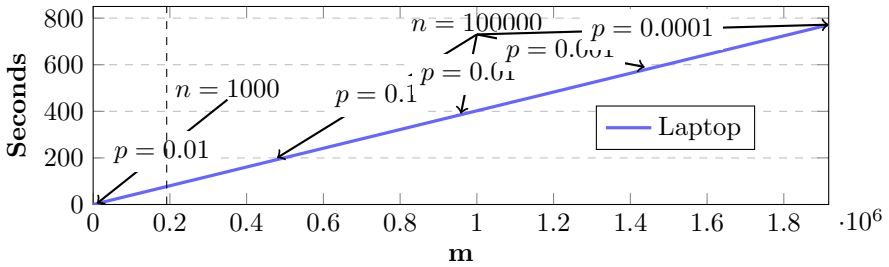


Figure 4.14: Consumer computation time. Dashed line is at m=191702; all computation times for n is 100, 1000 and 10000 are to the left of it, regardless which value of p we choose from Table 4.1.

lowest and approximately 12 minutes for the highest value of m. Additionally, we indicate on the graph the computation time for a setup with $n = 1000$ and $p = 0.01$, which is 3.8 seconds. We also indicate through a dashed line the value of m for which $n = 10000$ and $p = 0.0001$; all the parameter combinations except those with $n = 100000$ find themselves to the left of this line and lead to computation times lower than 77 seconds. Eventually, for $n = 100000$ we explicitly indicate the corresponding computation times.

4.5 Real-World Case Study: Assen TT festival

It is important to go beyond simulations and test our system using real-world data. Such data has some particularities that can be hardly reproduced in simulations. In a real-world setting, crowd sizes and directions of flows are generated by real people and their movements. The identifiers sensed by our system are, too, real, not synthetic. The distribution of the identifiers is unique, hard to mimic, as their appearance and occurrence is dictated by the devices and movements of the aforementioned real people, and it is also influenced by the particular setup in which the sensing takes place. In other words, facing our system with real-world data, reproducing the execution of it and analyzing the system's behavior is the closest we can get to a real deployment.

Every year, in the city of Assen, The Netherlands, a motorcycle grand prix[3] takes place which gathers crowds of people from all over Europe. In parallel, the municipality organizes the TT Festival[4] in the days before and after the race. The festival is spread across the city center, which is open only to pedestrians for the duration of the festival. There are concerts taking place on multiple stages, places for motorcycle stunts, street-food areas and amusement parks. Typically, more than a hundred thousand people visit this event, leading to a considerable amount of pedestrian movement between the attractions.

Figure 4.15: Placement of scanners in the city center of Assen.

Wi-Fi data gathering from scanners installed through the city was performed

[3]https://www.motogp.com/en/event/Netherlands
[4]https://ttfestival.nl/

during the 2015, 2016 and 2017 editions [61]. In this work, we use the dataset from 2017, when 30 Wi-Fi scanners were deployed in the city of Assen for 12 days, covering the whole period of the TT Festival as well as several days after the festival. In total, there were 26414742 detections of devices having 176888 different identifiers. A map with the placement of scanners is displayed in Fig. 4.15. Highlighted in red is the scanner placed in Koopmansplein, corresponding to the most visited area, having 1.6 million detections throughout the whole period. Also, we highlight in blue the scanners on Torenlaan street, which find themselves on the main path leading to a big parking on the edge of the city center and having a whole range of crowd flows happening between them.

For the setup phase, our system needs the following parameters: e_l - epoch length, n - a maximum number of devices expected near a scanner within the chosen epoch time, p - the probability of false positives when n devices are present. Being interested in capturing both footfall and crowd flow situations, we choose to fix e_l to 5 minutes. This is a time interval long enough to ensure that we capture probe requests from most of the devices broadcasting such messages near a certain scanner within an epoch, as well as short enough to interpret a detection of the same device at another scanner in a subsequent epoch as being part of a crowd flow. Analyzing the dataset, we see that for a 5 minutes long epoch most of the detection sets contain less than 1000 devices, with very few only marginally exceeding this threshold, which makes 1000 an appropriate setup value for n. We set p to 0.01 as we have seen from sections 4.3 and 4.4 that such value is expected to produce highly accurate responses and it is safely supported by the resource-constrained device that we use as scanner in our experiments.

We emulate the scanner in Koopmansplein by feeding the Raspberry Pi with detections from the dataset at the exact same pace as they happened in real life. We choose a continuous period spanning across 9 days, containing 3 festival nights followed by 6 festival-free days; this choice is consistent throughout the rest of the section, representing an interval in which all the concerned scanners (including those in Torenlaan street) uninterruptedly gathered detections. We allocate detections to corresponding epochs, according to the timestamps attached to them in the dataset, and we subject them to Algorithm 2 while using the same hash functions and encryption scheme as in the previous section. Getting as close as possible to reality, we want to test how well a scanner performs, as part of an actual implementation of our system, when faced with a real-world flow of detections. Also, we want to see how accurate the statistical counts of footfall queries are for such a setup. We are aware that based on Section 4.3 one can form an idea about what to expect; nevertheless, what we

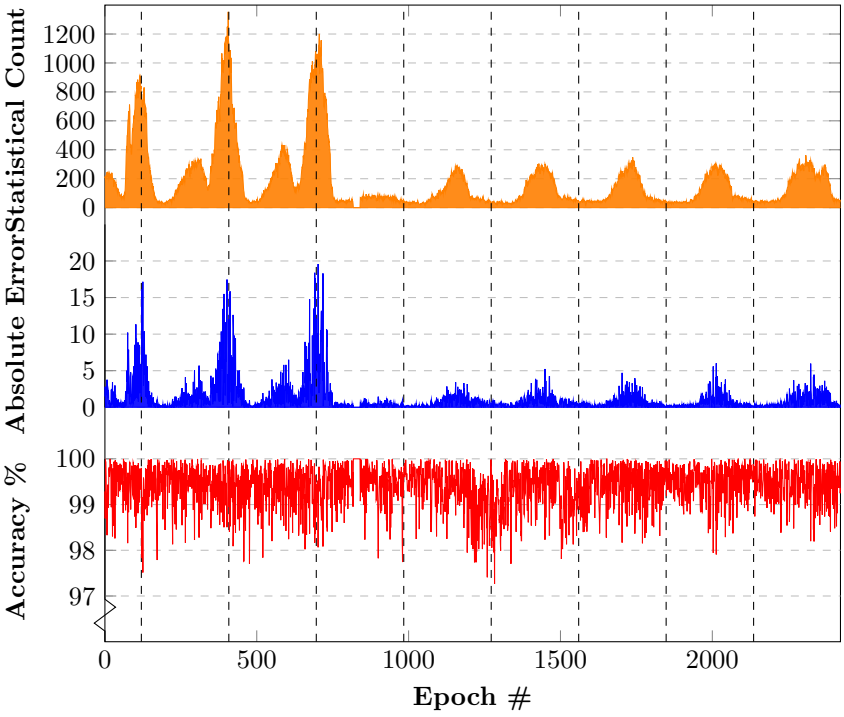


Figure 4.16: Estimating footfall in Koopmansplein during festival days and afterwards. Vertical dashed lines indicate midnight.

perform here is a neat reproduction of the festival environment as if our system was implemented there, dealing with real distributions of identifiers and real detections of them across time.

In Fig. 4.16 we plot the statistical counts estimated by a consumer based on the answers received from the system, the absolute errors of these counts (i.e. the absolute differences between real counts and estimations), as well as the accuracy according to the definition. The average processing time on the scanner was 12 seconds, with minor variations depending on footfall size. The highest absolute error is 19.58, where instead of 1012 devices the estimation was 992.42. The lowest observed accuracy is 97.2%, where instead of 36 devices the estimation was 35.01. The small gap in the fourth day corresponds to a period when readings did not reach the server, which is common across different scanners.

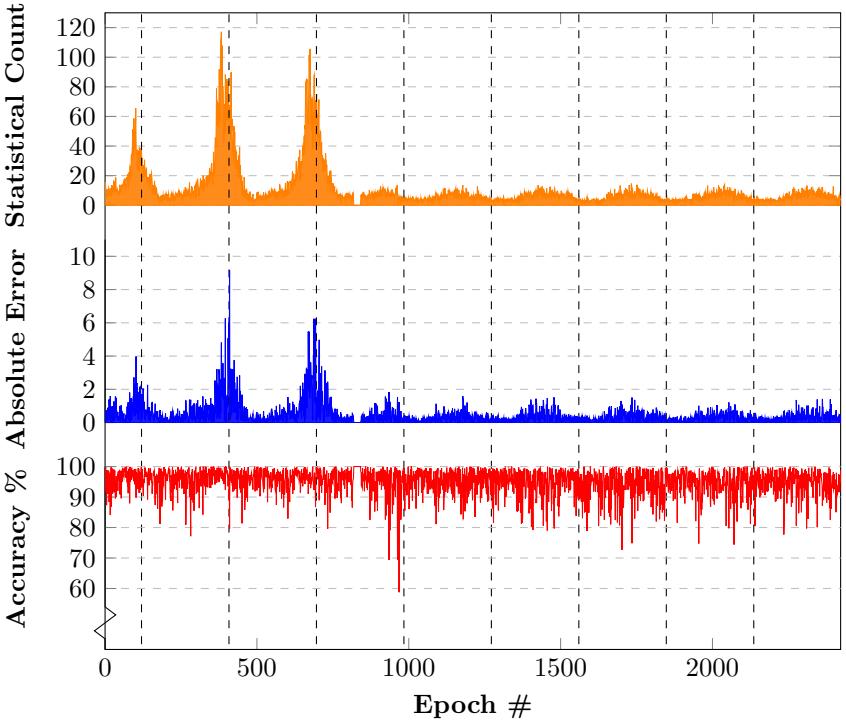


Figure 4.17: Estimating crowd flow size on Torenlaan street. Vertical dashed lines indicate midnight.

Using the same setup parameters, in Fig. 4.17 we have a look at the crowd flows between the two scanners in Torenlaan, flowing in the direction of the parking, which target devices making the transition between the two places in consecutive epochs. We plot the statistical counts a consumer estimates based on the results returned by our system, together with absolute errors, as well as the obtained accuracies. Please note that there is a one-to-one match between epochs in Fig. 4.17 and Fig. 4.16. The largest crowd flow has 122 devices in it, with an estimation of 117.18 and an accuracy of 96%. The lowest observed accuracy is 58.7% for a real count of 3 and an estimation of 4.23, though for 88.5% of the 2422 crowd flows the accuracy stays above 90%. The highest observed absolute error is 9.17, which happened for a real count of 45 estimated as 54.17. However, for 98.7% of the crowd flows, the estimation is

less than 3 devices away from the real count.

In Fig. 4.18 we group the crowd flows by their real count found on the x-axis. In the upper part we display the estimated counts as devices away from the real counts. For comparison, we show them overlapped with the graph from Fig. 4.9, which was plotting, as a worst-case scenario, the mean and standard deviation for crowd flows originating from crowds sized 1000. In the lower part we plot, for each real count, the mean accuracy of the estimated counts.

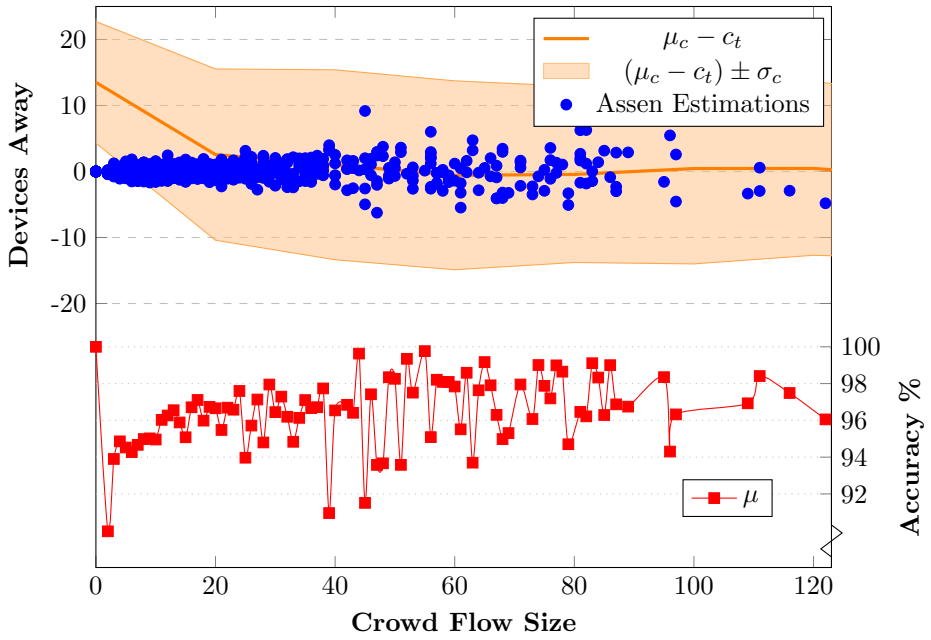


Figure 4.18: Estimating crowd flow size on Torenlaan street, grouped by real count and compared with Fig. 4.9.

All crowd flows happening on Torenlaan street are estimated within the boundaries of one worst-case standard deviation from the real count. Small crowd flows are much closer to the real count than to the expected mean as they come from much smaller initial crowds than they do in the worst-case; for Torenlaan street, crowd flows smaller than 10 usually result from intersecting initial crowds smaller than 100, compared to 1000 in the worst-case, systematically leading to less false matches and more accurate estimations. Accuracy-wise, the mean stays above 89.9% for all the encountered crowd flow

sizes. The decrease in smoothness once the crowd flow size increases comes from having fewer samples for larger crowd flows.

4.6 Discussion

4.6.1 Comparison with previous work

We have combined BFs with homomorphic encryption for computing statistical counts on crowds on another occasion [70], which we refer to as previous work throughout this section. The current chapter is also based on that paper, using the same building blocks but proposing a different construction. The main difference lies in where, when and how the query response is built, as well as in how the statistical count is calculated based on that response.

In our previous work the query response is built step by step *on the scanners* concerned by the query, each step a scanner carrying forward only the BF positions indicated by the hashes of identifiers sensed in that epoch, i.e. it multiplies the values found on those positions with an encrypted 1 and it writes an encrypted 0 on all the other positions. Eventually, the final scanner performs set membership testing under encryption, i.e. it multiplies for each sensed identifier the k hash-indicated positions together, assembling the results into a response for the consumer. Then, the consumer learns the statistical count by simply decrypting the response and counting the 1's.

In contrast, in current work the response is built *on the server*, whenever all the necessary data is available, by multiplying position-wise the encrypted BFs and shuffling the result; the statistical count is then estimated by the consumer by applying the proposed equations.

These construction differences have implications on three main dimensions of the system: performance, security and utility. Let us now detail each of them.

Performance. Involving scanners in the creation of responses, as we did in previous work, makes their load increase with the number of queries, eventually leading to a limitation in terms of queries a scanner can be involved into at the same time. Queries in crowd-monitoring systems come at varying pace and they can very well concern the same scanner in an overwhelming way, e.g., when a scanner is at a crossroad, being asked to produce data for numerous crowd flows passing through it. A scanner, which is a hardware device with fixed capabilities, is hard to scale in such settings. Moving most of the computationally expensive operations related to response creation on a server, as we do in current work, ties the load on the scanners to the number of enrolled consumers rather than

the number of queries. The number of enrolled consumers does not change often and, when it does, it is predictable. Thus, scanners have stable amounts of computation for long periods of time and it is known beforehand when they must scale. On the other hand, servers are more suitable for dealing with a varying pace of queries and they can scale much easier, as we have shown in 4.4.2.

Security. Scanners do not know any longer in which queries they are involved. They just perform sensing, write detections in BFs, encrypt and send them to the server, their role in the protocol being reduced. They do not receive any longer other encrypted BFs to perform operations on, which means a lower communication overhead as well as less communication rounds. The server gets more responsibility in the protocol, being trusted to correctly perform operations under encryption and shuffle results before sending them to consumers. Nevertheless, both variants of the protocol are secure under the same honest-but-curious adversary model.

Utility. In previous work, for data minimization purposes, the system was producing data necessary for responses only when it was required by a consumer through a query launched *before* the data collection had to start. While still being able to function this way if desired, the current system offers the additional possibility to launch queries concerning a certain situation also *after* it happened, as there are cases when interest arises post-factum, e.g., for investigating unexpected events. Another point on utility, the accuracy of the statistical counts differs between the two approaches, different ways of calculation being used. In general, it tends to be higher in previous work, as scanners, step by step, inherently drop detections that are not part of the intersection from BFs, this being an effect of their participation in the query response computation when their turn comes, which is no longer the case in current work.

4.6.2 Security Analysis

Our system, as we propose it, is secure against honest-but-curious adversaries, also known as semi-honest. Such adversaries follow the protocol correctly, but they may use the data they handle to learn more about the input of the other parties.

Honest-but-curious scanner. Scanners are assumed tamper-proof in the system model, otherwise there is no way to guarantee the proper functioning of the system. Nevertheless, dropping this assumption for a moment, by

compromising a scanner, an honest-but-curious attacker cannot learn more than the input of that scanner in the protocol, that is the detections made by it. It cannot learn detections of other scanners, as no information is exchanged among scanners, neither does it come from the server.

Honest-but-curious server. The server stores and processes EBFs. In order to see what is in there, it would need to be able to decrypt. Decryption is possible by using the private key of a consumer for whom the EBF is produced. Having that private key would mean that the SP colludes with that consumer, which would break the requirements of the system model. Another thing a server may try is encrypting 0's and 1's itself using public keys of consumers and compare them with encrypted values in EBFs, in an attempt to uncover the values stored under encryption. This would be meaningless due to the ciphertext indistinguishability property of the ElGamal scheme, also known as probabilistic encryption, which guarantees that encrypting the same value multiple times results in different ciphertexts.

Honest-but-curious consumer. A consumer can query the system and receive an EBF as answer. After decrypting the answer, knowing that BFs are used in the process, it may be tempted to check whether a certain device is in there by computing the hash functions on its identifier and searching for 1's in the corresponding positions. However, it would be useless since the EBF was shuffled before being sent to the consumer. Thus, the only meaningful information available after decryption is a statistical one, i.e. how many 0's and 1's are in there, which is all that a consumer needs to reach its goal according to the protocol, namely computing statistical counts.

The security of the system can be compromised in case the implementation of the system deviates from the system model. For example, if the non-colluding entities condition is not satisfied, an honest-but-curious server, even without being malicious, would be able to see what is stored in EBFs. Moreover, if the server is malicious, it could very well skip the shuffling part and deliver the EBFs as they are to the consumers.

4.7 Conclusion

In this work, we propose a novel crowd-monitoring system that produces statistical counts of crowds while fully protecting the privacy-sensitive data of the individuals being monitored. At the core of our solution lies a cryptographic construction in which the detections of individuals are encoded into homomorphically encrypted BFs and then immediately discarded. This construction

allows our system to blindly perform computations over encrypted data that it cannot decrypt, such that only statistical counts become available in the clear. Therefore, the system can accommodate, in a privacy-friendly way, footfall counting, as well as counting crowd flows between different locations, measurements otherwise unacceptable due to the risk of uniquely identifying individuals from the handled data.

We implement the system using Raspberry Pi as a scanner and different server configurations as operators under encryption. In addition to simulations, we test the system using real-world data from a large festival. For footfall scenarios concerning the most crowded area of the festival, the accuracy does not get below 97.2% (i.e. 1 device away in that particular case). Also, when measuring crowd flows happening on a circulated street, 88.5% of the statistical counts had an accuracy above 90%, 10.7% between 80% and 90%, 0.6% between 70% and 80% and 3 of them below 70%. For the same crowd flows, 98.7% of the estimations were less than 3 devices away from the real counts. These results demonstrate that highly accurate statistical counts of crowds are indeed practical when dealing with real-world data. Moreover, limited hardware (i.e. resource-constrained devices as scanners and a laptop as server) proves to be sufficient for this purpose, successfully accommodating a homomorphic encryption scheme on top of probabilistic data structures such as BFs. We hope that our work inspires other researchers searching for a solution in comparable settings, who aim to protect privacy-sensitive data sensed by an infrastructure of data collection points while still being able to use it for the intended purpose of their system.

Chapter 5

Anonymized Counting of Nonstationary Wi-Fi Devices

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. 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 [23] 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.

This chapter is based on the work presented in [73]. The rest of the chapter is structured as follows. Section 5.1 presents background information and reviews the related work. Section 5.2 introduces the system model. In Section 5.3 we propose our construction, followed by an evaluation in Section 5.4 and a discussion in Section 5.5. Finally, Section 5.6 concludes the chapter.

5.1 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 [67]. 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 [13, 69].

A common crowd-monitoring scenario is presented in Fig. 5.1. 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 work, can ask for more insights, such as how many of those devices displayed a nonstationary versus a stationary behavior.

Anonymization on the fly is a vital prerequisite for protecting collected data that is generated by individuals' devices from being exposed to privacy-invasive situations. In CMSs implementing this concept, such as [11] and also including the systems proposed in Chapters 3 and 4, 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, as we have shown in Chapter 4, i.e. 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

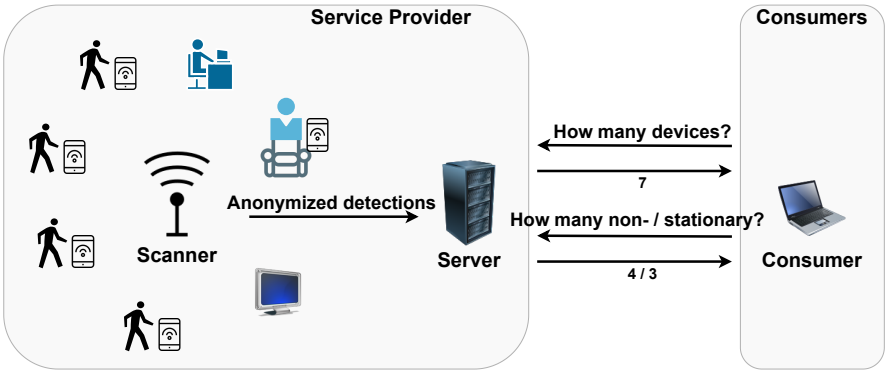


Figure 5.1: Intended behavior of a CMS offering statistical counts on footfall, including the ability to count separately nonstationary and stationary devices.

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. [25] investigated the use of so-called *stay points* [48]. 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 [62]. 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 [10] 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 be assigned by manufacturers to devices of both types, thus making the approach impractical for our purpose.

5.2 System Model

5.2.1 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. 5.1). 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.

5.2.2 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 13. 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 14. 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]

[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

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 chapter 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 15. 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.

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.

5.2.3 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 try to infer as much insights as possible from the data handled by that component. 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, nor 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

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.

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.

5.3 Our Construction

In this section 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. The privacy protection part is ensured by the same building blocks used in the method proposed in Chapter 4.

5.3.1 Statistical counting with Bloom filters

For computing statistical counts on crowds, we have proposed, in Chapter 4, to make use of Bloom filters (BFs) [18]. We recall the characteristics of BFs: m - BF length (i.e. number of positions in the array), k - number of hash functions associated with a BF, p - probability of false positives expected when n elements are present in the BF.

We leave security aside for a moment to recall the functionality under the hood. We will get back to this in subsection 5.3.3, where we present the complete multi-party cryptographic construction, 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 [74], by computing the formula in eq. (5.1). This estimation is highly accurate, as shown by Papapetrou et al. in [59]. Please note that this is the same formula we have used for estimating footfall in Chapter 4.

$$c = -\frac{m}{k} \ln\left(1 - \frac{c_t}{m}\right) \tag{5.1}$$

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

5.3.2 Combing: Separately counting nonstationary from stationary devices

Estimating statistical counts using eq. (5.1) 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 15, and write them in two different BFs. Then, the corresponding granular statistical counts could be computed by separately applying eq. (5.1) 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 16. 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* [39].

We present in Fig. 5.2 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. (5.1) generates the estimated counts nc and sc, which approximate the statistical counts $|\mathcal{ND}_{s,e}|$ and $|\mathcal{SD}_{s,e}|$.

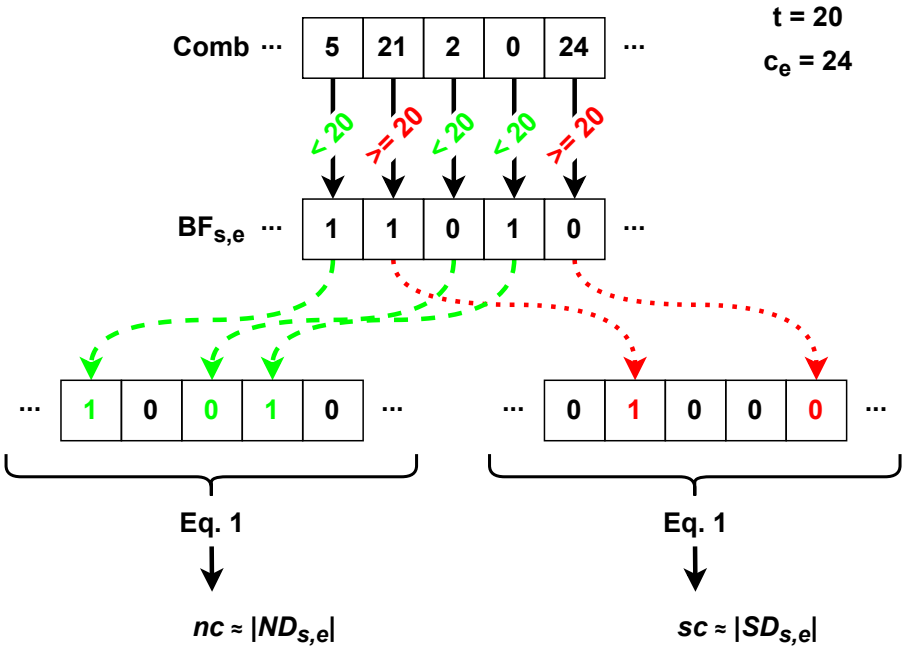


Figure 5.2: Combing a BF for $t = 20$, $c_e = 24$, in order to compute the statistical counts of nonstationary and stationary devices seen by scanner s during epoch e.

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.

5.3.3 Anonymized counting under encryption

The representation of data in BFs, despite being different from detections in the clear, still leaves the stored 0's and 1's visible. As the identifier space (i.e. MAC address space) is easily enumerable [16], 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.

In Chapter 4, to enable combining BFs under encryption for counting crowd flows, we used homomorphic encryption [64], i.e. a type of encryption that allows mathematical operations directly on the encrypted data, without the need for decryption. More precisely, we used the multiplicatively homomorphic version of ElGamal [37]. In particular, in this system, in order to build the comb, we need an encryption scheme allowing additions under encryption, so we opt to use the additively homomorphic version of ElGamal.

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. (5.1) to estimate the statistical counts.

5.4 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.

5.4.1 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 [12], 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. 5.3, 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).

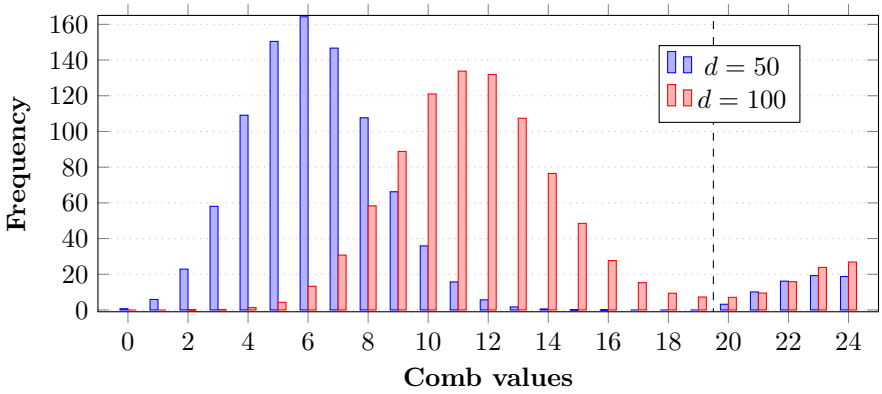


Figure 5.3: Frequency of comb values when sensing 50 and 100 devices per epoch for $c_e = 24$ epochs, using BFs configured for $n = 100$ and $p = 0.01$. The dashed line marks the threshold $t = 20$.

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 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.

5.4.2 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 5.4.4, 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. (5.1) 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. 5.4.

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.

(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,

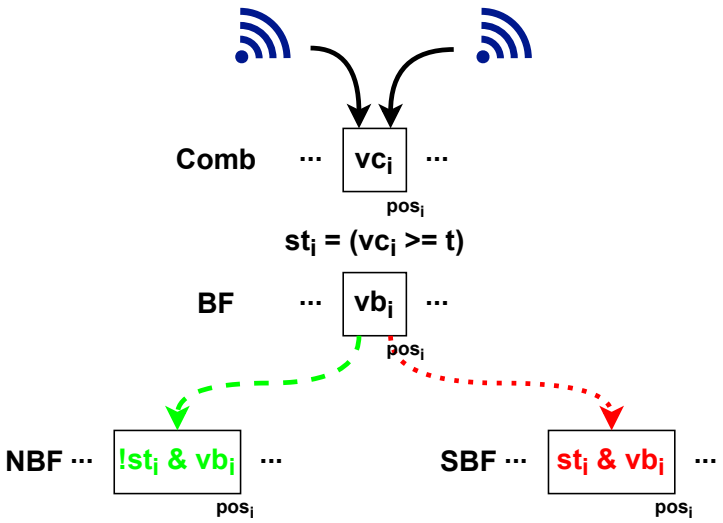


Figure 5.4: Moving value vb_i from a colliding position i in a BF to the nonstationary (NBF) or stationary (SBF) BF-like structure, based on the relationship between its corresponding value in the comb vc_i and t.

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. (5.2).

$$p' = \left(1 - \left(1 - \frac{1}{m}\right)^k\right)^2 \tag{5.2}$$

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.

5.4.3 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. As mentioned in Chapter 4, 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 [41]. 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 5.4.2. We will later show in Section 5.4.4 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 15). 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. 5.5, the split between nonstationary and stationary devices counted by using our mechanism, when setting the threshold t to 20, 10 and 5, respectively.

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.

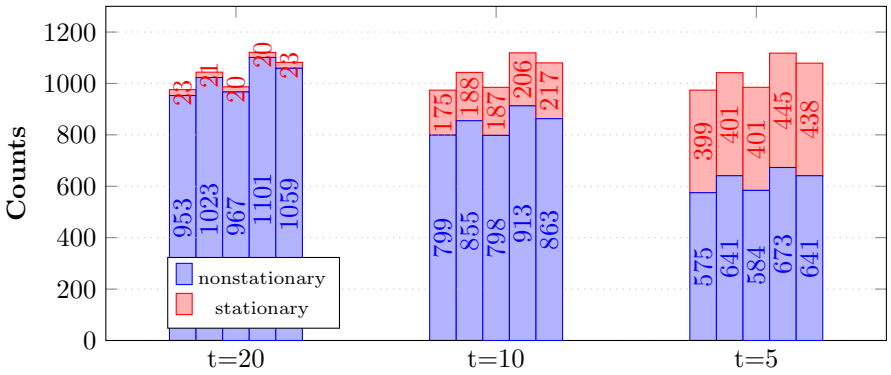


Figure 5.5: Separate counts of nonstationary and stationary devices for $c_e = 24$ and different values of t.

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 5.5.1.

For the same sequence of epochs, we want to see how accurate the split we have just presented is. We plot thus in Fig. 5.6, 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) shows stationary devices.

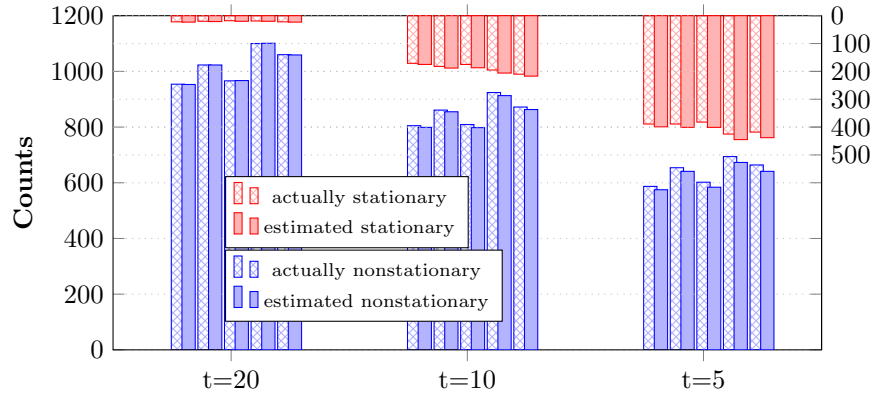


Figure 5.6: Comparing real counts of nonstationary and stationary devices with counts estimated by the system.

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 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. 5.7, the absolute error of nonstationary and stationary counts when t equals 20, 10 and 5.

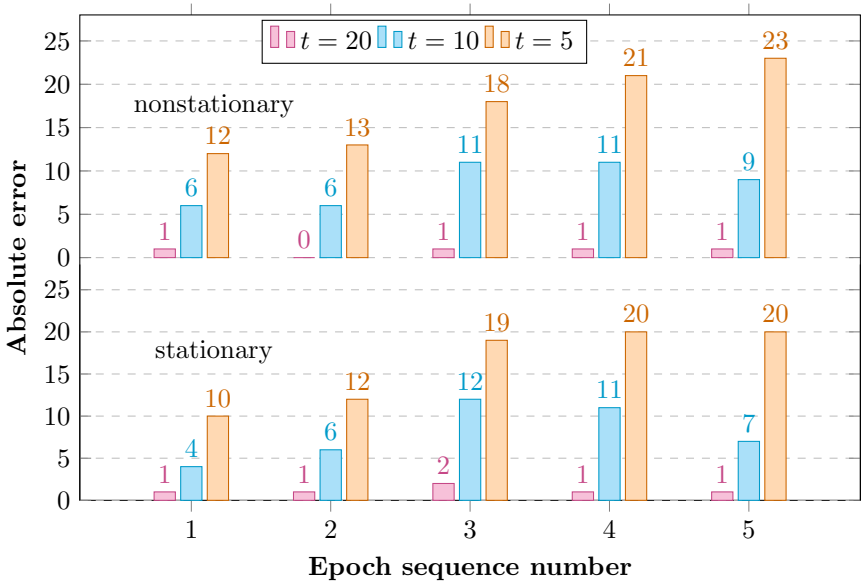


Figure 5.7: Absolute error of counts of nonstationary and stationary devices.

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. 5.8, 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.

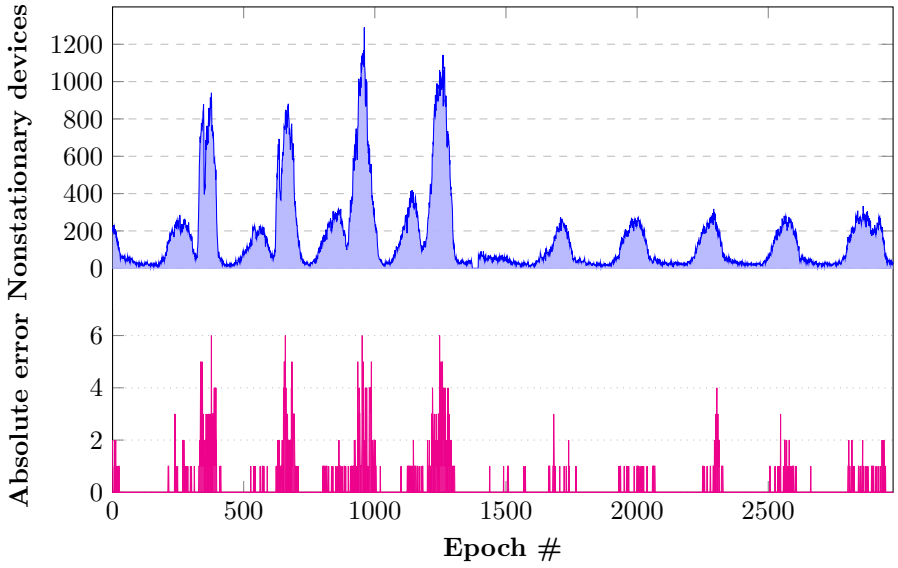


Figure 5.8: Estimated nonstationary devices and their absolute errors during festival days and afterwards.

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.

5.4.4 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 [9] and use the SCAPI[2] library [35] 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

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

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.

5.5 Discussion

5.5.1 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.

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.

5.5.2 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,[3] as we present below.

Detections are encoded into BFs and then immediately discarded at the 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 through 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.

[3] HBC adversaries do not deviate from the protocol but try to learn as much as possible from the data they handle.

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.

5.5.3 Limitations on number of consumers

Homomorphic encryption is known to be resource-demanding. With our resources, we were able to set up the system in such a way 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.

5.5.4 Integration with crowd flows counting

The capability to separately count nonstationary from stationary devices is intended to be integrated in a CMS such as the one presented in Chapter 4. The privacy protection guarantees are similar for the two works, making the protocols designed to achieve these guarantees compatible with each other and, therefore, enabling a handy attachment of this novel capability to such CMS. However, the used variants of ElGamal homomorphic encryption are slightly different. More precisely, the way in which 0's and 1's are encoded under encryption differs in order to accommodate two different types of operations, i.e. additions for building a comb and multiplications for combining EBFs to count crowd flows. Thus, in order to support footfall and crowd flow counting at the same time with separately counting nonstationary from stationary devices, scanners should build, for each epoch and for each enrolled consumer, two EBFs having their positions encoded according to the two different ElGamal variants. The BF parameters of the two EBFs, though, do not necessarily have to be identical, allowing the SP to set them separately.

5.6 Conclusion

In this chapter, 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, in 93.9% of the estimation 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.

Chapter 6

Conclusion

Wi-Fi-based crowd-monitoring systems are nowadays widely deployed, offering valuable insights on the dynamics developing within crowds of pedestrians. At the same time, such insights are built upon detections of devices of people making up these crowds, people who have privacy concerns. These systems have been initially built without taking privacy of individuals into account and existing attempts to patch them with privacy-protecting capabilities proved to be insufficient on numerous occasions.

In this thesis we aimed to protect the privacy of individuals when their devices are sensed by a Wi-Fi-based crowd-monitoring system. Also, we wanted to do this in such a way that the functionality of the system remains valid. Hence, we proposed, as our main research question, to find out how to construct Wi-Fi-based crowd-monitoring systems in such a way that they enable discovering pedestrian dynamics and protect the privacy of the sensed individuals at the same time. To address the main research question, we split the research in three research questions: RQ1 on suitable techniques for privacy protection, RQ2 on the utility level achievable using these privacy protection techniques and RQ3 on the efficiency of the proposed methods. Below we will show how these questions have been responded through the contributions presented in the chapters of the thesis.

6.1 Contributions

RQ1: What techniques can be used for managing privacy-sensitive crowd-monitoring data such that *privacy protection* is ensured?

We proposed two methods to ensure privacy protection for crowd-monitoring

data. While protection is ensured by both methods, the type of offered protection differs. One of them is based on k-anonymity and it was presented in Chapter 3, while the other one combines Bloom filters with homomorphic encryption, being displayed in Chapter 4. Both of them perform anonymization on the fly, i.e. data is anonymized directly, in isolation, before leaving the scanners. The first one ensures that no matter how the data is combined after anonymization in order to address crowd-monitoring queries, it is impossible to have an identifier appearing less than k times. The second one drops entirely the idea of using identifiers by encoding detections into Bloom filters and encrypting them with a homomorphic encryption scheme, leaving no possibility of going back to the original data under the proposed cryptographic construction.

RQ2: To what extent do the considered privacy protection techniques impact the *utility level* of the attainable crowd-monitoring insights?

The proposed methods both allow the computation of statistical counts on crowds, i.e. measuring the footfall in a location as well as the size of a crowd flow between locations. The impact of both privacy protection methods on measuring footfall proved to be low. However, the utility level differs when it comes to crowd flows, as the k-anonymity approach proved to achieve less utility than the Bloom filters approach in the following sense. The accuracy of the counts on detection k-anonymous crowd flows decreases when the number of people joining and leaving the crowd increases, limiting thus its applicability to situations known to have few perturbations. On the other hand, the Bloom filters approach is not sensitive to such variations, confirmed by achieving highly accurate estimations for most situations when experimenting with a wide range of real-world crowd flows, i.e. 98.7% of the estimations being at most 3 devices away from the real counts.

Obtaining such results with the Bloom filters approach made us look into whether we can enable the option of making granular decisions upon which devices to be considered as part of the crowd, willing to expand even further the utility of statistical counts. Thus, we explored in Chapter 5 the possibility of separately counting nonstationary from stationary devices under the same privacy protection guarantees as in Chapter 4, an attempt which succeeded, achieving a mean accuracy of 99.9% while experimenting with the same real-world dataset.

RQ3: How *expensive* are the considered privacy protection techniques for crowd monitoring from an *efficiency* point of view?

To answer this question, we performed proof-of-concept implementations of the systems and assessed their performance. The k-anonymity approach does not incur a significant cost on the performance of the system, as the

applied operations are lightweight, truncation implying a binary operation and correction (if needed) a sorting algorithm. The Bloom filters approach is more expensive, as cryptographic operations, which tend to be heavy, are performed in the process. However, our implementation, which used a resource-constrained device as a scanner, and first a laptop then a cloud environment as a server, could successfully accommodate several consumers enrolled at the same time, being able to perform the necessary operations below the allowable time limit needed to ensure the liveness of the system.

6.2 Limitations & Directions for Future Research

In the following paragraphs we will reflect upon the results of our research from the perspective of encountered boundaries, thus discussing limitations of our work, as well as presenting potential avenues for future research starting from our findings.

6.2.1 Detection k-anonymity's sensitivity to perturbations

In Chapter 3 we realized that despite being highly efficient, detection k-anonymity is sensitive to perturbations, i.e. the percentage of people joining or leaving a crowd flow. For crowd flows known to have few perturbations, the accuracy is high. If there are more perturbations, the accuracy decreases. Our intuition made us ask ourselves whether knowing information on the percentage of expected joiners and leavers can help getting better crowd flow estimations. To quickly test this, we performed a preliminary implementation of a multiple linear regression algorithm and the improvements indeed looked promising for the presented London Underground routes. However, more research is needed to get a complete picture of all the implications.

6.2.2 Clarifications on statistical counts

Wi-Fi scanners cover a certain range where they gather detections from. This range is influenced by the physical characteristics of the installation location and *it fluctuates in time* because there are phenomena that are known to influence Wi-Fi signals propagation, such as reflection, refraction, scattering, diffraction, absorption or shadowing. So when we say that a device is near a scanner, it means that the device is in the range of the scanner at a specific moment. However, being in the range of a scanner does not guarantee that a device is detected. For a detection to happen, a device should broadcast a Wi-Fi probe request message within that epoch (sending patterns differ across

devices) and that message should reach the scanner (packet loss can occur due to congestion or interference).

Literally, the footfall as we model in our work is the number of *distinct identifiers* in the Wi-Fi messages reaching a scanner within an epoch. This number is different from the number of people in that area, as there are people that carry no device while others carry more than one device. Moreover, this is even different from the number of detected devices, as there might be devices that use MAC address randomization and re-randomize their broadcasted identifier in a period of time shorter than the epoch duration. Also, it is worth mentioning that this number is independent of how many times the identifiers are seen, since we made the decision to use *detection sets*.

Continuing along the same lines, a statistical count on a crowd flow is the number of distinct identifiers found in the Wi-Fi messages that reach the concerned scanners during the specific epochs indicated by the crowd flow query. In addition to the influences mentioned for footfall, the closeness of crowd flows to real-life flows of people is influenced by the distance between scanners. Measuring crowd flows between distant scanners must take into account the traveling speed of pedestrians, such that appropriate epochs are chosen. On the other hand, crowd flows between close scanners must take into account the possibility of detecting the same device in multiple places at the same time, because overlapping ranges, though not desirable for crowd-monitoring, can happen in practice.

6.2.3 Dealing with overlapping ranges of scanners

Throughout the thesis we made the assumption that, ideally, scanners are positioned in such a way that their ranges do not overlap. In real-world implementations, though, it can happen that scanners overlap in coverage, either temporarily, due to the fluctuation of ranges, or by design, for contiguous coverage purposes. By following the definition, a crowd flow between such scanners will falsely count devices that find themselves in the overlap as making the transition. The problem is most impactful for crowd flows of complexity $q = 2$, i.e. between two scanners, as having at least a third one would already indicate an actual movement unless all of them overlap, which is uncommon.

In Fig. 6.1, for scanners with overlapping ranges s_1, s_2 and epochs e_1, e_2, we present the 4 possible situations in which devices might find themselves when counted as part of the crowd flow. All situations show devices detected by s_1 in e_1 and by s_2 in e_2. Situation (a) is similar with the case when scanners have non-overlapping ranges, device a_1 being detected by a single scanner each epoch. In (b) and (c), in one of the epochs devices are detected by both

scanners, being in the overlap of ranges. Eventually, devices like a_4 in (d) are detected by both scanners in both epochs.

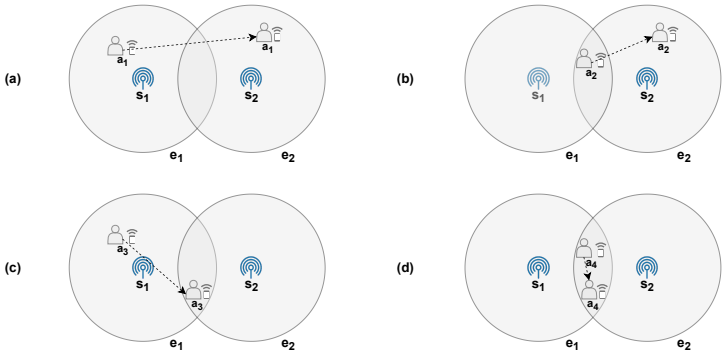


Figure 6.1: Situations of crowd flows between scanners with overlapping ranges.

An improved estimation of the crowd flow in discussion could be achieved, for example, by computing $|\mathcal{D}_{s_1,e_1} \cap \mathcal{D}_{s_2,e_2}| - |\mathcal{D}_{s_1,e_1} \cap \mathcal{D}_{s_2,e_2} \cap \mathcal{D}_{s_1,e_2} \cap \mathcal{D}_{s_2,e_1}|$. This formula excludes from counting devices in (d), for which no information regarding their movement can be derived. Other strategies, such as excluding devices in (b) or (c), can be applied in the same way.

Both systems proposed in Chapters 3 and 4 can support such calculations. For example, the system proposed in Chapter 4 can achieve this by asking the server to perform the additional necessary BF intersections under encryption, then shuffle and deliver the result to the consumer along with a query response. We tested this on crowd flows between two scanners from the Assen dataset that seemed to have overlapping ranges and indeed we were able to identify and subtract apparently unmoving devices from the statistical counts. To what extent, though, the improved statistical counts are closer to the actual flows cannot be deduced from the available data only.

6.2.4 Crowd flows between more than two locations

In Chapter 4, when analyzing the accuracy of crowd flows, we essentially looked at combining two BFs and used an improved closed formula [59] for estimating the counts. However, there is no improved closed formula for combining more than two BFs and applying the general (unimproved) formula makes accuracy quickly decrease.

To accurately estimate the counts for such crowd flows (i.e. concerning more than two locations), there are currently two available options that could

be investigated. The first one is applying the general estimation formula that was used also for footfall. Note, though, that BFs may need to be designed larger in order to maintain high accuracy for the more crowded flows. The second one is resorting to the variant proposed in [70], that does not have the accuracy negatively affected by the number of combined BFs. Yet, it should be dealt with its particularities, such as the fact that queries must be specified beforehand and heavy computations have to be performed on the scanners.

Bibliography

[1] https://marketinglaw.osborneclarke.com/advertising-regulation/jc-decauxs-pedestrian-tracking-system-blocked-by-french-data-regulator/, 2015. [Online; accessed 04-June-2020].

[2] https://www.gizmodo.co.uk/2017/02/heres-what-tfl-learned-from-tracking-your-phone-on-the-tube/, 2017. [Online; accessed 04-June-2020].

[3] http://content.tfl.gov.uk/review-tfl-wifi-pilot.pdf, 2017. [Online; accessed 04-June-2020].

[4] https://www.rtvoost.nl/nieuws/303852/Enschede-stopt-tijdelijk-met-wifi-tellingen-na-publicatie-Autoriteit-Persoonsgegevens, 2018. [Online; accessed 04-June-2020].

[5] https://www.autoriteitpersoonsgegevens.nl/nl/nieuws/bedrijven-mogen-mensen-alleen-bij-hoge-uitzondering-met-wifitracking-volgen, 2018. [Online; accessed 04-June-2020].

[6] https://autoriteitpersoonsgegevens.nl/nl/nieuws/boete-gemeente-enschede-om-wifitracking, 2021. [Online; accessed 18-February-2022].

[7] N. Abedi, A. Bhaskar, and E. Chung. Bluetooth and wi-fi mac address based crowd collection and monitoring: Benefits, challenges and enhancement. In *Australasian Transport Research Forum 2013 Proceedings*, pages 1–17. Australasian Transport Research Forum, 2013.

[8] O. Abul, F. Bonchi, M. Nanni, et al. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, volume 8, pages 376–385, 2008.

[9] M. Adalier and A. Teknik. Efficient and secure elliptic curve cryptography implementation of curve p-256. In *Workshop on Elliptic Curve Cryptography Standards*, volume 66, 2015.

[10] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren. Usage patterns in an urban wifi network. *IEEE/ACM Transactions on Networking*, 18(5):1359–1372, 2010.

[11] M. Alaggan, M. Cunche, and S. Gambs. Privacy-preserving wi-fi analytics. *Proceedings on Privacy Enhancing Technologies*, 2018(2):4–26, 2018.

[12] A. Appleby. Murmurhash3. 2016.

[13] L. Bai, N. Ireson, S. Mazumdar, and F. Ciravegna. 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*, pages 432–440, 2017.

[14] A. Basalamah. Crowd mobility analysis using wifi sniffers. *Int. J. Adv. Comput. Sci. Appl*, 7(12):374–378, 2016.

[15] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Workshop on Secure Data Management*, pages 185–199. Springer, 2005.

[16] G. Bianchi, L. Bracciale, and P. Loreti. " better than nothing" privacy with bloom filters: To what extent? In *International Conference on Privacy in Statistical Databases*, pages 348–363. Springer, 2012.

[17] U. Blanke, G. Tröster, T. Franke, and P. Lukowicz. Capturing crowd dynamics at large scale events using participatory gps-localization. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–7. IEEE, 2014.

[18] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[19] E. D. P. Board. Opinion 05/2014 on anonymisation technique. `https://www.pdpjournals.com/docs/88197.pdf`, 2014. [Online; accessed 04-June-2020].

[20] B. Bonné, A. Barzan, P. Quax, and W. Lamotte. Wifipi: Involuntary tracking of visitors at mass events. In *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2013.

[21] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang. On the false-positive rate of bloom filters. *Information Processing Letters*, 108(4):210–213, 2008.

[22] J. Cao, B. Carminati, E. Ferrari, and K. L. Tan. Castle: A delay-constrained scheme for k s-anonymizing data streams. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1376–1378. IEEE, 2008.

[23] Z. Cao, Y.-E. Sun, H. Huang, H. Guo, Y. Du, A. Liu, and L. Lu. Finding persistent elements of anomalous flows in distributed monitoring systems. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE, 2020.

[24] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences*, 231:83–97, 2013.

[25] C. Chilipirea, C. Dobre, M. Baratchi, and M. van Steen. Identifying movements in noisy crowd analytics data. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, pages 161–166. IEEE, 2018.

[26] M. Cunche. I know your mac address: Targeted tracking of individual using wi-fi. *Journal of Computer Virology and Hacking Techniques*, 10(4):219–227, 2014.

[27] M. Cunche, M.-A. Kaafar, and R. Boreli. Linking wireless devices using information contained in wi-fi probe requests. *Pervasive and Mobile Computing*, 11:56–69, 2014.

[28] L. Demir, M. Cunche, and C. Lauradoux. Analysing the privacy policies of wi-fi trackers. In *Proceedings of the 2014 workshop on physical analytics*, pages 39–44. ACM, 2014.

[29] L. Demir, A. Kumar, M. Cunche, and C. Lauradoux. The pitfalls of hashing for privacy. *IEEE Communications Surveys & Tutorials*, 20(1):551–565, 2017.

[30] N. Doulamis. Evacuation planning through cognitive crowd tracking. In *2009 16th International Conference on Systems, Signals and Image Processing*, pages 1–4. IEEE, 2009.

[31] A. Draghici and M. V. Steen. A survey of techniques for automatically sensing the behavior of a crowd. *ACM Computing Surveys (CSUR)*, 51(1):21, 2018.

[32] M. Dunlap, Z. Li, K. Henrickson, and Y. Wang. Estimation of origin and destination information from bluetooth and wi-fi sensing for transit. *Transportation Research Record*, 2595(1):11–17, 2016.

[33] C. Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[34] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[35] Y. Ejgenberg, M. Farbstein, M. Levy, and Y. Lindell. Scapi: The secure computation application programming interface. *IACR Cryptol. ePrint Arch.*, 2012:629, 2012.

[36] J. El Mallah, F. Carrino, O. Abou Khaled, and E. Mugellini. Crowd monitoring. In *International Conference on Distributed, Ambient, and Pervasive Interactions*, pages 496–505. Springer, 2015.

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

[38] European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation), 2016.

[39] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM transactions on networking*, 8(3):281–293, 2000.

[40] E. Fenske, D. Brown, J. Martin, T. Mayberry, P. Ryan, and E. Rye. Three years later: A study of mac address randomization in mobile devices and when it succeeds. *Proceedings on Privacy Enhancing Technologies*, 2021(3):164–181, 2021.

[41] J. Freudiger. 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*, pages 1–6, 2015.

[42] Y. Fukuzaki, M. Mochizuki, K. Murao, and N. Nishio. Statistical analysis of actual number of pedestrians for wi-fi packet-based pedestrian flow sensing. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 1519–1526. ACM, 2015.

[43] A. Guillén-Pérez and M. D. C. Baños. A wifi-based method to count and locate pedestrians in urban traffic scenarios. In *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 123–130. IEEE, 2018.

[44] D. Helbing, P. Molnár, I. J. Farkas, and K. Bolay. Self-organizing pedestrian movement. *Environment and planning B: planning and design*, 28(3):361–383, 2001.

[45] H. Hong, C. Luo, and M. C. Chan. Socialprobe: Understanding social interaction through passive wifi monitoring. In *Proceedings of the 13th international conference on mobile and Ubiquitous systems: Computing, networking and services*, pages 94–103, 2016.

[46] A. Johansson, D. Helbing, H. Z. Al-Abideen, and S. Al-Bosta. From crowd dynamics to crowd safety: a video-based analysis. *Advances in Complex Systems*, 11(04):497–527, 2008.

[47] M. Kamp, C. Kopp, M. Mock, M. Boley, and M. May. Privacy-preserving mobility monitoring using sketches of stationary sensor readings. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 370–386. Springer, 2013.

[48] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello. Extracting places from traces of locations. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(3):58–68, 2005.

[49] C. E. Kontokosta and N. Johnson. Urban phenology: Toward a real-time census of the city using wi-fi data. *Computers, Environment and Urban Systems*, 64:144–153, 2017.

[50] J. Li, B. C. Ooi, and W. Wang. Anonymizing streaming data for privacy protection. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1367–1369. IEEE, 2008.

[51] J. Ma, W. Song, S. M. Lo, and Z. Fang. New insights into turbulent pedestrian movement pattern in crowd-quakes. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(02):P02028, 2013.

[52] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24. IEEE, 2006.

[53] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown. A study of mac address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies*, 2017(4):365–383, 2017.

[54] M. Marx, E. Zimmer, T. Mueller, M. Blochberger, and H. Federrath. Hashing of personally identifiable information is not sufficient. *SICHERHEIT 2018*, 2018.

[55] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz. Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society B: Biological Sciences*, 276(1668):2755–2762, 2009.

[56] A. Musa and J. Eriksson. Tracking unmodified smartphones using wi-fi monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*, pages 281–294, 2012.

[57] M. E. Nergiz, M. Atzori, and Y. Saygin. Towards trajectory anonymization: a generalization-based approach. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, pages 52–61. ACM, 2008.

[58] F. of Privacy Forum. `https://fpf.org/wp-content/uploads/10.22.13-FINAL-MLA-Code.pdf`, 2013. [Online; accessed 04-June-2020].

[59] O. Papapetrou, W. Siberski, and W. Nejdl. Cardinality estimation and dynamic length adaptation for bloom filters. *Distributed and Parallel Databases*, 28(2):119–156, 2010.

[60] M. Perttunen, V. Kostakos, J. Riekki, and T. Ojala. Urban traffic analysis through multi-modal sensing. *Personal and Ubiquitous Computing*, 19(3-4):709–721, 2015.

[61] A.-C. Petre, C. Chilipirea, M. Baratchi, C. Dobre, and M. van Steen. Wifi tracking of pedestrian behavior. In *Smart Sensors Networks*, pages 309–337. Elsevier, 2017.

[62] A. E. C. Redondi, D. Sanvito, and M. Cesana. 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*, pages 51–58, 2016.

[63] G. D. P. Regulation. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016.

[64] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[65] F. Rusu and A. Dobra. Statistical analysis of sketch estimators. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 187–198, 2007.

[66] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, technical report, SRI International, 1998.

[67] L. Schauer, M. Werner, and P. Marcus. 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*, pages 171–177. ICST (Institute for Computer Sciences, Social-Informatics and ..., 2014.

[68] B. Soundararaj, J. Cheshire, and P. Longley. Estimating real-time high-street footfall from wi-fi probe requests. *International Journal of Geographical Information Science*, pages 1–19, 2019.

[69] B. Soundararaj, J. Cheshire, and P. Longley. Estimating real-time high-street footfall from wi-fi probe requests. *International Journal of Geographical Information Science*, 34(2):325–343, 2020.

[70] V.-D. Stanciu, M. v. Steen, C. Dobre, and A. Peter. Privacy-preserving crowd-monitoring using bloom filters and homomorphic encryption. In *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*, pages 37–42, 2021.

[71] V.-D. Stanciu, M. van Steen, C. Dobre, and A. Peter. Privacy-friendly statistical counting for pedestrian dynamics. Under review.

[72] V.-D. Stanciu, M. van Steen, C. Dobre, and A. Peter. k-anonymous crowd flow analytics. In *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 376–385, 2020.

[73] V.-D. Stanciu, M. van Steen, C. Dobre, and A. Peter. 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 on International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 213–222, 2022.

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

[75] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[76] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*, volume 8, pages 65–72, 2008.

[77] P. Torkamandi, L. Kärkkäinen, and J. Ott. An online method for estimating the wireless device count via privacy-preserving wi-fi fingerprinting. In *International Conference on Passive and Active Network Measurement*, pages 406–423. Springer, 2021.

[78] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens. Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 413–424. ACM, 2016.

[79] K. A. Wallace. Anonymity. *Ethics and Information technology*, 1(1):21–31, 1999.

[80] J. Weppner, B. Bischke, and P. Lukowicz. Monitoring crowd condition in public spaces by tracking mobile consumer devices with wifi interface. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 1363–1371. ACM, 2016.

[81] M. Wirz, T. Franke, D. Roggen, E. Mitleton-Kelly, P. Lukowicz, and G. Tröster. Inferring crowd conditions from pedestrians' location traces for real-time crowd monitoring during city-scale mass gatherings. In *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 367–372. IEEE, 2012.

[82] M. Wirz, E. Mitleton-Kelly, T. Franke, V. Camilleri, M. Montebello, D. Roggen, P. Lukowicz, and G. Troster. Using mobile technology and a participatory sensing approach for crowd monitoring and management during large-scale mass gatherings. In *Co-evolution of Intelligent Sociotechnical Systems*, pages 61–77. Springer, 2013.

[83] Y. Yuan. Crowd monitoring using mobile phones. In *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 1, pages 261–264. IEEE, 2014.

[84] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia. Continuous privacy preserving publishing of data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 648–659. ACM, 2009.