Privacy-Aware Crowd Monitoring and WiFi Traffic Emulation for Effective Crisis Management

Privacy-Aware Crowd Monitoring and WiFi Traffic Emulation for Effective Crisis Management

Riccardo Rusca[†‡], Alex Carluccio[†], Diego Gasco[†], Paolo Giaccone[*‡]

[†] Department of Control and Computer Engineering, Politecnico di Torino, Italy
[*] Department of Electronics and Telecommunications, Politecnico di Torino, Italy
[‡] Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Parma (PR), Italy

***Abstract*—Estimating the number of people in a given area, denoted as "people counting" process, plays a vital role in crisis management and disaster response, enabling accurate monitoring of crowd dynamics and facilitating effective decision-making. In this work, we focus on WiFi fingerprint technique which exploits the MAC address of the mobile devices as proxy for people counting. Due to the European GDPR regulation and the strict actions undertaken by the major smart-devices vendors to enhance users' privacy (e.g., MAC randomization), most of the techniques investigated in the past must be redesigned and rethought. Here, we propose an ad-hoc WiFi traffic generator, tailored to emulate a realistic behaviour of the WiFi cards and to provide the ground truth for the counting algorithms. Furthermore, we propose a technique for crowd monitoring that leverages Bloom filters to guarantee a formal "deniability" property, which preserves users' privacy. Our solution is also compatible with trajectory-based crowd monitoring.**

I. Introduction

In crisis management and disaster response scenarios, accurate people counting and crowd monitoring play a pivotal role in facilitating effective decision-making and ensuring public safety. The ability to precisely assess crowd dynamics, estimate resource requirements, and optimize emergency response efforts is crucial for authorities. However, the task of counting and tracking individuals in large gatherings or chaotic situations has never been an easy one. Traditional techniques, such as surveillance cameras, LiDAR and infrared systems, WiFi and Bluetooth fingerprints, have been extensively utilized but they are now facing challenges due to the recent European General Data Protection Regulation (GDPR) [1] and heightened user privacy concerns by major smart-devices vendors.

This paper focuses on addressing the evolving landscape of people counting and crowd monitoring techniques in light of the GDPR and the need for enhanced privacy protection. Specifically, we consider the approach based on counting WiFi probe requests and we propose an ad-hoc probe request generator which emulates in details real-world behavior patterns observed in our recent work [2]. The traces obtained by our generator can be used as ground truth for counting algorithms, allowing their enhancement and fine tuning.

Furthermore, we present a novel crowd-monitoring technique that effectively utilizes probe request messages in conjunction with Bloom filters. By leveraging the formal deniability property, defined γ-deniability in [3], we demonstrate the preservation of users' privacy, making our solution GDPR compliant.

In summary, this paper addresses the challenges posed by GDPR and the evolving landscape of user privacy concerns in the context of people counting and crowd-monitoring. We propose a one-of-a-kind probe request generator and a privacy-preserving crowd monitoring technique, based on a novel idea of "anonymization noise". We aim to contribute to the development of effective solutions that enable accurate monitoring of crowd dynamics while upholding the highest standards of privacy protection.

The rest of the paper is organized as follows. Section II discusses some relevant related work. Section III describes the realistic probe request generator. Section IV presents the Bloom filter system used to store data in a privacy preserving manner. Section V describes our experimental settings to derive some metrics that enable us to validate the proposed generator, and it shows the effectiveness of our solution to infer the trajectory of flows of people. Finally, Section VI draws our conclusions.

II. Related works

In the last few years crisis management has underlined an important factor in emergency events: crowd estimation. Indeed, having an estimate of how many people are present in a certain environment and how they are moving, can be very useful to manage in a better way each situation. During the last few years, several techniques have been proposed to address the challenge of people counting. In [4], the authors discuss the problem of people counting in multi-camera surveillance systems. The proposed method combines partial body detection and person re-identification to accurately count the number of people in the overlapping area. Similarly, in [5] the authors explain a system for detecting and counting people using computer vision techniques, focusing on overhead view-based detection. Both the previous articles demonstrated good results, but, on the other hand, techniques leveraging video camera have several problems. First of all, the high hardware cost due to the high resource-computation demand required, secondly, outdoor scenarios are very challenging due to light variations and in presence of large density of people; finally, recording and storing face detection is subject to privacy issues. Furthermore, recent works [6], [7] exploited the use of LiDAR sensors, which compared to video camera techniques solve the privacy issues, but still the hardware cost and the environment use case remain a problem. Conversely, the

979-8-3503-1951-4/23/$31.00 ©2023 IEEE

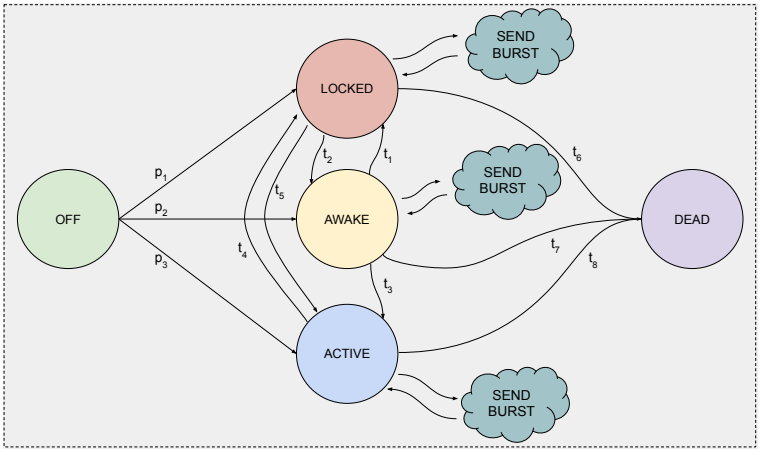

Fig. 1: State-machine probe requests generator diagram

works in [8] and [9] analyze the use of WiFi probe request messages as a method to monitor crowds in various scenarios by collecting WiFi fingerprints of mobile devices. Solutions that leverage such messages can be used in both indoor and outdoor scenarios, they require low-budget equipment, low computational requirements, and can tackle privacy issues. At the same time, WiFi fingerprints have some drawbacks, primarily related to the lack of ground truth data, needful to test and fine-tune counting algorithms.

Regarding the recent advancements in terms of European regulations, the GDPR [1] heavily restricts the storage and handling of sensitive data. As mentioned earlier, mobile devices emit frequent probe request messages, containing information (such as the MAC address), useful for device identification and monitoring. *MAC addresses are classified as personal data by the GDPR, for this reason, they must be subject to privacy protection mechanisms [10].* Several solutions have been presented in the literature to address this issue, including [11], [12]. Both of these solutions address the privacy problems by leveraging Bloom filters to store MAC addresses information and by using an asymmetric homomorphic encryption system, applied to the Bloom filter data. The homomorphic encryption preserves the possibility of computing intersection between different Bloom filters, allowing the computation of crowd flow trajectories.

As demonstrated in [3], Bloom filters can be used to preserve privacy of stored data, but to achieve a formal level of anonymity required by the GDPR they require some minimum number of data insertions. In few words, few inserted data do not guarantee anonymity. This observation is not considered in [11], [12], thus their proposed solutions become valid only for enough large crowd. Furthermore, [3] introduces two fundamental concepts for anonymity protection: γ-deniability and γ-K-anonymity. The first property tells us that an element stored in the Bloom filter is *deniable* if it can be replaced by other elements not originally inserted into the filter without changing the Bloom filter bitmap. The second property extends the first by stating that an element in a Bloom filter is γ-K-anonymous if the element can be "covered", with a probability of γ, by $K-1$ other elements not originally inserted into the filter. We will exploit the concept of γ-deniability, which is nicely tailored to GDPR requirements, in Section IV.

Our work differs from the prior art as we focused on creating a probe requests generator addressing the lack of datasets with the related ground truth data. Moreover, we analyzed and proposed a GDPR-compliant solution to store users' sensitive data allowing at the same time the use of people counting and crowd estimation algorithms.

III. Realistic generator of probe request traces

To properly design a realistic probe request generator, we started by understanding the probe request behaviour, how their sending process is implemented by different device vendors in current generation smartphones, and how the user interaction with the device itself influences the sending of probe request messages. We sum up all the work done in [2]. Starting from the learned knowledge, we created a state-machine probe request generator, as the one in Figure 1, to have tunable traces as ground truth, where it would be possible to vary the number of devices and their models as desired and then validate counting algorithms.

It is noteworthy to highlight that upon powering on a device, we assign three probabilities for it to initiate in any of the three potential phases. Additionally, following a timeout, the device can undergo a state change, either transitioning to a different phase or entering a dead phase, indicating that the device is once again turned off. During the Locked, Awake, and Active states, the device has the capability to enter a transient state called "send burst", wherein it emits a burst and subsequently returns to its previous state. In [2] we underlined how each smartphone has a different way of acting in probe request sending, depending on vendor implementation and user interaction with the device. For this reason, we decided to extract and save for each device the following metrics:

- whether a randomized MAC address is used;
- number of packets inside a burst (burst length);
- 802.11 VHT capabilities;
- 802.11 Extended capabilities;
- 802.11 HT capabilities;
- time between packets inside the same burst (inter-packet time);
- time between different bursts (inter-burst time).

Since 2014, device vendors have been implementing MAC address randomization to protect user privacy [13]. Coherently, in our past study [2], we observed MAC randomization, especially for more recent devices. Based on our findings, we opted to utilize a fully randomized MAC address, except for the observed devices where only the second half of the MAC address is modified while retaining the first 3 bytes (i.e., the OUI). Moreover, in order to generate random MAC addresses, we set the global/local bit (the second-least-significant bit in the first byte) to 1 and we randomly choose all the other values. Regarding the VHT and Extended capabilities, as well as burst length, inter-packet time, and burst rate statistics, we created a database of value-probability pairs, divided for each device state (i.e., locked, awake, and active), coherently with what observed in [2]. The state machine is realized as a scheduling queue, where each event has its starting time, and the queue is ordered based on that time. The events that are handled inside the generator are the following:

- *CreateDevice*: event that creates a new object of type Device. The initial phase of the device is decided with an experimentally chosen probability distribution. The same event is responsible for the creation of other related events, such as *DeleteDevice*, *ChangePhase*, *CreateBurst*, and *CreateDevice*. To decide which vendor and model of the device to create with the latter event, we referred to [14].
- *DeleteDevice*: an event that deletes the device object and all the events associated with it that are in the queue, if any.
- *ChangePhase*: event that chooses the next device phase, based on the current one and on the aforementioned probabilistic distribution. Then we schedule the events *ChangePhase* and *CreateBurst*.
- *CreateBurst*: event that creates several events of type *SendPacket* according to the device's probabilistic distribution of the burst length value.
- *SendPacket*: event that contains the built probe request packet and saves it to the output `.pcap` file.

In order to maintain a consistent presence of a specific number of devices within the simulated environment, it is possible to configure two parameters in the simulator: the average number of devices and the duration of their presence in the area. To set the proper arrival rate of new devices, we adopted the well-known Little's Formula:

$$L = \lambda W \tag{1}$$

where L represents the average number of customers in a system (i.e., devices in the area) at any given time, λ represents the average arrival rate of customers into the system (i.e., arrival rate of new devices) and W represents the average time a customer spends in the system (i.e., duration of device presence). Following this law, we can compute λ and schedule the arrival of the next device according to a Poisson process, while maintaining a given average number of devices in our generator.

An open-source implementation of the complete probe request generator has been made accessible to the research community[1], facilitating the effortless and practical generation of new datasets.

TABLE I: Notations and definitions

Notation	Definition
U	Set of elements in the universe
S	Set of elements stored in the Bloom filter
$BF(S)$	Bloom filter storing a set S
$n = \|S\|$	Number of elements stored in the Bloom filter
m	Size in bit of the Bloom filter
k	Number of hash functions
t_i	Number of bits set to 1 in the Bloom filter BF_i
V	Hiding Set

IV. Bloom filters for people counting

We decided to leverage Bloom filters for people counting and crowd monitoring. However, we faced two primary challenges: ensuring privacy preservation and enabling the identification of people's movement patterns.

In order to provide some background, we remind that a Bloom filter is a probabilistic data structure used to represent a set of elements. It is implemented using an array of bits $BF \in \{0,1\}^m$ of length m and k independent hash functions $H_1 ... H_k$, that maps an input x to one of the m bits of the bit array. Let $BF[i]$ be the ith bit of BF. Initially, all bits are set to 0. To insert an element x into the Bloom filter, the k hash functions are applied to x, and the bits of BF corresponding to the generated positions from the hash functions are set to 1:

$$BF[H_i(x)] = 1 \qquad \forall i = 1, \ldots, m \tag{2}$$

In order to verify if an element is present in the Bloom filter, the element is hashed through to the same k functions, and the output is compared to the current values of the corresponding bits in BF. If the 1s in the output match the corresponding bits in BF (i.e., both are set to 1), the element is considered "probably present" in the Bloom filter. However, if even a single bit in the match is set to 0, the element is considered definitely not present in the Bloom filter. It is important to notice that a Bloom filter can potentially provide false positives, meaning that it may mistakenly indicate that an element is present even if it is not. However, it does not provide false negatives, meaning that it cannot mistakenly

[1] https://github.com/riccardo-rusca/ProbeRequestGenerator

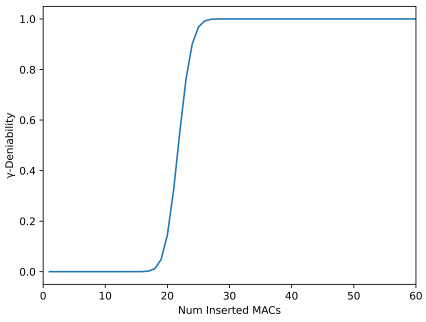

Fig. 2: γ-deniability value in relation to the number of inserted MAC addresses.

indicate that an element is not present when it actually is. Based on the available memory m, it is possible to compute the optimal value of k such that a given value of probability of false positive is achieved.

We now describe how Bloom filters can be used to preserve anonymity, as proposed in [3]. Before going further, we need to introduce some definitions, still derived from [3]:

- Set $S \subseteq U$ of elements stored in the Bloom filter from an universe U.
- Bloom filter $BF(S)$ storing S.
- Hiding set: $V = \{v_i\}$ is defined as *hiding set* for $BF(S)$ if it contains all the elements $v_i \in U$ such that v is not stored in $BF(S)$ (i.e., $v_i \notin S$) and a query for v_i returns true, i.e., present in $BF(S)$.
- Deniable: An element $x \in S$ inserted in $BF(S)$ is defined *deniable* if $\forall i \in \{1 \dots k\}$ it exists at least one hiding set element $v \in V$ for which $\exists j \in \{1 \dots k\}$ such that $H_i(x) = H_j(v)$. In other words, an element is deniable if it can be replaced by another element not inserted in the Bloom filter without changing the bitmap.
- γ-deniability: A $BF(S)$ is *γ-deniable* whenever a randomly chosen element $x \in S$ is deniable with probability γ.

In order to manage the privacy concern in accordance with the General Data Protection Regulation [1], *we propose to employ γ-deniability property as proposed in [3], specifically with $\gamma = 1$* to ensure that all the elements in the Bloom filters (i.e., the stored MAC addresses) are deniable.

As shown in [3], given a Bloom filter it is possible to compute γ for the level of γ-deniability according to:

$$\gamma(BF(S)) = \left(1 - \exp\left(-\frac{hk}{m(1 - e^{-nk/m})}\right)\right)^k \quad (3)$$

where n is the number of stored elements in the Bloom filter, and $h = (|U| - n)(1 - e^{-nk/m})^k$ is the average hiding set cardinality and $|U|$ is $2^{48} \approx 2.8\ 10^{14}$.

Figure 2 shows the level of deniability after inserting n MAC addresses, according to (3), for $m = 10,000$ bits and $k = 7$ hash functions. It can be shown that the chosen value of k is the optimal one minimizing the probability of false positive when $n = 1,000$ elements are stored. From the figure, it can be seen that after 30 insertions in the Bloom filter, the value of γ reaches 1, thus providing the certainty that there will be always at least one element belonging to the hiding set that can be used to deny an element inserted in the Bloom filter.

We thus propose to define an *anonymization noise* composed of $n_{\min}$ elements randomly generated to be inserted into the Bloom filter as soon as it is instantiated. In the scenario of Figure 2, $n_{\min} = 30$. This will provide the ability to guarantee 1-deniability and thus ensure the anonymity of the data from the first insertion of a MAC address into the Bloom filter.

Consider now the scenario in which many WiFi counting sensors are located in a large area, and each sensor stored the MAC addresses in a local Bloom filter. By knowing the common MAC addresses between different Bloom filters, it is possible to identify the people's movement patterns. Fortunately, it is to possible to compute approximately the

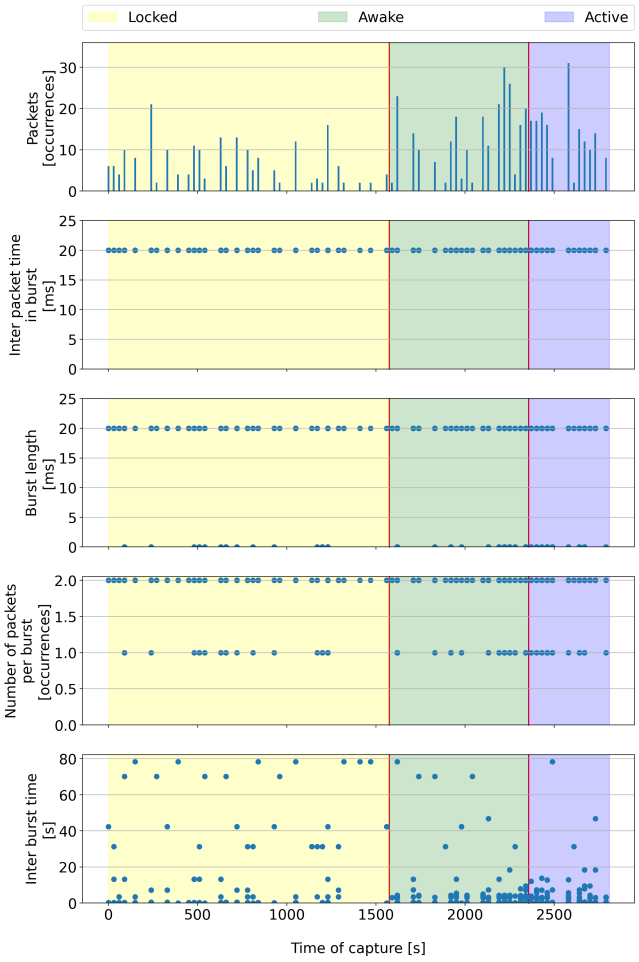

Fig. 3: Trace of emulated bursts for Apple iPhone 11 as a function of time, for different device phases. Different colored bands are used to highlight the different phases of the experiment, according to the legend on top.

intersection of different Bloom filters by computing a bitwise *AND* operation between Bloom filters. To estimate the number of elements present in the intersection as accurately as possible, we compared two formulas. Assume to have two Bloom filters BF_1 and BF_2. Let $BF_3 = BF_1$ AND BF_2 be their intersection. Let t_k be the number of ones in the Bloom filter k: $t_k = \sum_{i=1}^{m} BF_k[i]$. The simplest way to estimate the number of MAC addresses in BF_3 is to apply the following formula, well known [15]:

$$c_1 = -\frac{m}{k} \ln\left(1 - \frac{t_3}{m}\right) \quad (4)$$

but it is approximated due to the way BF_3 is obtained.

A more accurate formula for the intersection BF_3 was shown in [16]:

$$c_2 = \frac{\ln\left(m - \frac{t_3 \times m - t_1 \times t_2}{m - t_1 - t_2 + t_3}\right) - \ln(m)}{k \times \ln(1 - \frac{1}{m})} \quad (5)$$

V. Numerical Evaluation

We first present in Sec. V-A the results validating our realistic probe request generator, then in Sec. V-B we investigate the most accurate way to estimate crowd trajectories.

TABLE II: Locked phase for Apple iPhone 11: (mean, coefficient of variation)

Metric	Real	Simulation 1	Simulation 2	Simulation 3	Simulation 4
Packets [occurrences]	(3.11, 2.52)	(3.42, 1.33)	(3.32, 1.36)	(3.29, 1.29)	(3.25, 1.34)
Inter packet time [ms]	(20.38, 0.02)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)
Burst length [ms]	(16.3, 0.52)	(16.73, 0.44)	(16.5, 0.46)	(16.76, 0.44)	(16.85, 0.43)
Packets per burst [occurrences]	(1.8, 0.23)	(1.84, 0.2)	(1.83, 0.21)	(1.84, 0.2)	(1.84, 0.2)
Inter burst time [s]	(16.49, 1.62)	(16.07, 1.59)	(16.48, 1.57)	(16.74, 1.56)	(17.01, 1.55)

TABLE III: Awake phase Apple iPhone 11: (mean, coefficient of variation)

Metric	Real	Simulation 1	Simulation 2	Simulation 3	Simulation 4
Packets [occurrences]	(7.54, 1.46)	(7.46, 1.15)	(7.74, 1.18)	(7.79, 1.13)	(7.42, 1.11)
Inter packet time [ms]	(20.33, 0.02)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)
Burst length [ms]	(17.94, 0.37)	(16.68, 0.45)	(16.55, 0.46)	(16.66, 0.45)	(16.5, 0.46)
Packets per burst [occurrences]	(1.88, 0.17)	(1.83, 0.2)	(1.83, 0.21)	(1.83, 0.2)	(1.83, 0.21)
Inter burst time [s]	(7.2, 2.32)	(7.35, 2.26)	(7.06, 2.33)	(7.04, 2.3)	(7.36, 2.27)

TABLE IV: Active phase Apple iPhone 11: (mean, coefficient of variation)

Metric	Real	Simulation 1	Simulation 2	Simulation 3	Simulation 4
Packets [occurrences]	(11.16, 1.06)	(10.47, 0.89)	(10.85, 0.91)	(10.87, 0.89)	(10.71, 0.85)
Inter packet time [ms]	(20.27, 0.02)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)	(20.0, 0.0)
Burst length [ms]	(16.6, 0.47)	(16.4, 0.47)	(16.85, 0.43)	(16.62, 0.45)	(16.61, 0.45)
Packets per burst [occurrences]	(1.82, 0.21)	(1.82, 0.21)	(1.84, 0.20)	(1.83, 0.2)	(1.83, 0.21)
Inter burst time [s]	(4.81, 2.41)	(5.2, 2.39)	(5.08, 2.42)	(5.03, 2.38)	(5.11, 2.27)

A. *Generator of realistic probe request traces*

The experiments done with the generator of realistic probe request traces are based on a series of simulations, each one producing an output trace with a fixed duration in terms of simulated time. We report in Figure 3 the results relative to a single device, the Apple iPhone 11. Each simulation is performed in a closed environment, where only the device considered "working" is active in our generator, any other device is added during the entire simulation. The objective of the generator is to emulate the real traces with a device's behavior that follows the real one, for this reason, in order to validate the probe request traces generated, we compared several metrics extracted from the real traces (derived from [2]) and the simulated ones.

Figure 3 shows the main statistics relative to only one simulation instance. With three different color bands, we represent the different phases in which the device is sending probe request at a certain time, the yellow band refers to the Locked phase, the green band to the Awake phase, and the purple band to the Active phase. The entire simulation lasts 40 minutes and in each graph, we considered an observation window of 30 seconds. Notably, it took less than 6 seconds to generate the whole trace, showing the scalability of the proposed approach.

From the top down, the first plot shows the number of packets generated in the observation window. Compliant with the real traces and the results shown in [2], within the Locked phase, the device tends to send more packets at the beginning than at the end, on the other hand within the Awake and Active phases the number of packets generated increases. The second plot shows the time between packets inside the same burst, called henceforth *InterPacketTime*. It can be shown that the results are perfectly matching the real ones. The third plot reports the burst duration, i.e., the difference between the arrival time of the last packet and the first packet inside the same burst. This value is strictly correlated to the previous one and it can be seen that simulate very well the real behavior (it is worth mentioning that the zero values refer to the burst with one packet only). The fourth plot shows the number of packets per burst. Finally, the last plot represents the time between the capture of the last packet of a burst and the capture of the first packet in the next burst, called henceforth *InterBurstTime*. Once again both the results obtained from the generated traces of the last two plots are perfectly following the real traces.

To extend our validation, we decided to generate more traces from our generator and extract, for each of the five metrics described in Figure 3, the mean and the coefficient of variation evaluated among all the simulations. Moreover, we compared the simulated values with the real values extracted from the real probe request traces of the considered device. We sum up all the results in Tables II, III, and IV, reporting the comparison between the real trace metrics and the simulated ones, for three different phases. Four simulations have been done with

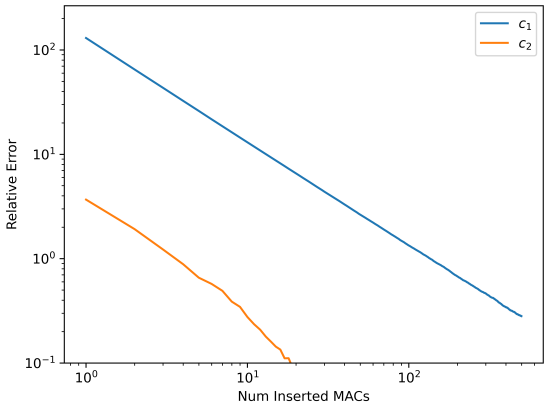

Fig. 4: Relative error of the estimators c_1 and c_2 to evaluate the number of people moving from one scanner to the other.

a simulated time equal to three hours, one hour per phase. With these long simulations, we were able to show that the first two moments of the simulated behavior of the device is matching very well the real behavior of the device. This is thanks to an internal procedure of the simulator which generates the messages according to the empirical distributions measured in the real traces.

B. Evaluation of crowd trajectories

We run some simulated experiments in which we took two Bloom filters, BF_1 and BF_2, with $10,000$ bit each, and $k = 7$ hash functions. We generated 500 distinct MAC address and inserted each of the Bloom filters, to model the background presence of devices which appear individually in each Bloom filter, corresponding to people that passed just under one WiFi scanner. To model people moving from one scanner to another, we added one by one a common MAC address in both BF_1 and BF_2. Then, after each insertion, we computed the intersection in BF_3 and used (4) for c_1 and (5) for c_2 to estimate the number of common MAC addresses in the two Bloom filters, which correspond to the people moving from the area around one scanner to the area around the other one.

Figure 4 shows the relative error after the insertion of the MAC address of the people moving from one scanner to the other. The relative error of c_1 appears to be very large for small number of stored MAC addresses, and becomes acceptable ($< 100\%$) only for more than 100 MAC addresses are stored. Instead, the relative error of c_2 is very small also for few stored MAC addresses. This is not surprising, since coherent with [16], but highlights a different tradeoff between accuracy, complexity and privacy to compute the number of MAC addresses in the intersection of BF_1 and BF_2. Indeed, (4) requires only to know the number of ones in BF_3, whereas (5) (even if very accurate) requires to know the number of ones in BF_1, BF_2 and BF_3 in the same server. Thus, the two solutions have different levels of adaptability to a given architecture, being either centralized or distributed. We leave for future work the exploration of the best tradeoff between accuracy, implementation complexity and privacy-preserving architectures.

VI. Conclusions

We designed a WiFi traffic generator emulating with high accuracy the behavior of real devices in generating the probe request messages. The generator is able to create realistic traffic traces lasting tens of minutes in few seconds for a single device. It is possible to generate any given set of devices at the same time, allowing to emulate the behavior of a large number of devices (also hundreds). This achievement has some practical relevance since it allows to create a ground truth scenario in which the number of devices is known a priori and on which it is possible to test the accuracy of different counting algorithms based on Probe Requests.

Furthermore, we exploited Bloom filter to preserve privacy guaranteeing the 1-deniability property thanks to the introduction of anonymization noise, for which we provided the methodology to set it properly. We also evaluated the effect of different formulas, taken from the literature, to estimate the number of people moving from one WiFi scanner to another, when the corresponding MAC addresses are stored in privacy preserving Bloom filters.

Both contributions provide the ground to design new counting algorithms able to cope with MAC randomization and with privacy concerns.

Acknowledgments and Disclaimer

This Work is funded by the European Union's Horizon-JU-SNS-2022 Research and Innovation Programme through the TrialsNet project (Grant Agreement No. 101095871). This manuscript reflects only the authors' views and opinions, and do not necessarily reflect the view the European Union neither the European Commission can be considered responsible for them.

References

[1] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council." [Online]. Available: https://data.europa.eu/eli/reg/2016/679/oj

[2] R. Rusca, F. Sansoldo, C. Casetti, and P. Giaccone, "What WiFi probe requests can tell you," in *IEEE Consumer Communications & Networking Conference (CCNC)*, 2023, pp. 1086–1091.

[3] G. Bianchi, L. Bracciale, and P. Loreti, ""Better than nothing" privacy with bloom filters: To what extent?" in *Privacy in Statistical Databases*. Springer Berlin Heidelberg, 2012.

[4] L. Ding, S. Wang, R. Li, L. Chen, and J. Dong, "PC-PINet: Partial re-identification network for people counting with overlapping cameras," in *International Conference on Image, Vision and Computing (ICIVC)*, 2021, pp. 66–71.

[5] E. P. Myint and M. M. Sein, "People detecting and counting system," in *IEEE Global Conference on Life Sciences and Technologies (LifeTech)*, 2021, pp. 289–290.

[6] A. Günter, S. Böker, M. König, and M. Hoffmann, "Privacy-preserving people detection enabled by solid state LiDAR," in *International Conference on Intelligent Environments (IE)*, 2020, pp. 1–4.

[7] Z. Chen, W. Yuan, M. Yang, C. Wang, and B. Wang, "SVM based people counting method in the corridor scene using a single-layer laser scanner," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 2632–2637.

[8] K. Gebru, M. Rapelli, R. Rusca, C. Casetti, C. F. Chiasserini, and P. Giaccone, "Edge-based passive crowd monitoring through WiFi beacons," *Computer Communications*, vol. 192, pp. 163–170, 2022.

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

[10] Preliminary verification. collection, analysis and processing of data, through the installation of equipment, for marketing and market research purposes. [Online]. Available: https://www.garanteprivacy.it/home/docweb/-/docweb-display/docweb/9022068

[11] V.-D. Stanciu, M. v. Steen, C. Dobre, and A. Peter, "Privacy-preserving crowd-monitoring using Bloom filters and homomorphic encryption," in *International Workshop on Edge Systems, Analytics and Networking (EdgeSys)*. ACM, 2021, p. 37–42.

[12] V.-D. Stanciu, M. van Steen, C. Dobre, and A. Peter, "Anonymized counting of nonstationary Wi-Fi devices when monitoring crowds," in *International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. ACM, 2022, p. 213–222.

[13] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. Rye, and D. Brown, "A study of MAC address randomization in mobile devices and when it fails," *Proceedings on Privacy Enhancing Technologies*, 2017.

[14] "MS Windows NT kernel description," https://gs.statcounter.com/vendor-market-share/mobile/europe/yearly-2020-2023-bar.

[15] "Bloom filter." [Online]. Available: https://en.wikipedia.org/wiki/Bloom_filter

[16] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for Bloom filters," *Distributed and Parallel Databases*, vol. 28, pp. 119–156, 2010.