Privacy-preserving WiFi-based Crowd Monitoring

Riccardo Rusca[†‡], Alex Carluccio[†], Claudio Casetti[†‡], 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*—The process of estimating the number of individuals within a defined area, commonly referred to as *people counting*, is of paramount importance in the realm of safety, security and crisis management. It serves as a crucial tool for accurately monitoring crowd dynamics and facilitating well-informed decision-making during critical situations. In our current study, we place a special emphasis on the utilization of the WiFi fingerprint technique, leveraging probe request messages emitted by smart devices as a proxy for people counting. However, it is essential to recognize the evolving landscape of privacy regulations and the concerted efforts by major smart-device manufacturers to enhance user privacy, exemplified by the introduction of MAC addresses randomization techniques. In this context, we designed a crowd monitoring solution that exploits Bloom filters for ensuring a formal *deniability*, aligning with the stringent requirements set forth by regulations like the European GDPR [1]. Our proposed solution not only addresses the essential task of people counting but also incorporates advanced privacy-preserving mechanisms. Importantly, it seamlessly integrates with trajectory-based crowd monitoring, offering a comprehensive approach to managing crowds while respecting individual privacy rights.**

***Keywords*—Crowd Monitoring, People counting, WiFi, Probe Request, Bloom filter, Anonymization noise**

I. Introduction

In the aftermath of the COVID-19 pandemic, social distancing measures have reshaped our world, leading to unprecedented restrictions on public gatherings. Even with the subsiding pandemic, limitations on gathering sizes continue to affect public events. Despite these challenges, we anticipate a resurgence of large public events in the post-COVID era, heralding excitement and also posing formidable challenges. The return of these mass gatherings introduces significant security and congestion management challenges. In this context, the role of crowd management analytics is essential for facilitating effective decision-making and ensuring public safety. The ability to meticulously assess crowd dynamics, estimate resource requirements, and optimize emergency response efforts is critical for authorities tasked with managing large public events. Nevertheless, the task of accurately counting and tracking individuals within large-scale gatherings remains a complex challenge. Traditional techniques, including the use of surveillance cameras, LiDAR and infrared systems, as well as WiFi and Bluetooth fingerprint tracking, have been extensively employed for this purpose. Nonetheless, these methods are now grappling with a set of new challenges, most notably those stemming from the European General Data Protection Regulation (GDPR) [1] and heightened concerns regarding user privacy, especially from prominent smart device manufacturers. Apart from pandemic-related concerns, crowd monitoring is crucial for ensuring safety during large events for several reasons. It enables authorities to anticipate and mitigate potential security threats, manage crowd flows to prevent stampedes or congestion, and allocate resources effectively in emergencies. In the event of an unforeseen crisis, such as a terrorist threat or a natural disaster, accurate people counting and crowd tracking can be instrumental in orchestrating rapid evacuations and providing timely medical assistance.

In this study, we harness IoT technology to address the challenge of simultaneously characterizing flows of people and quantifying the number of devices/people in a crowd, all while safeguarding user privacy.

Within our proposed crowd counting framework, we have harnessed a versatile data structure capable of facilitating both single scanner counting and flow analysis involving multiple scanners. When handling data captured from the WiFi scanners, our process begins with an initial stage of outlier removal. Subsequently, we leverage a derandomization technique, as proposed in [2], to address the randomization of MAC addresses. This technique allows us to assess the probability that different probe requests, each with distinct randomized MAC addresses, belong to the same device. This step significantly enhances the counting accuracy. The output of the derandomization process is then stored in a Bloom filter, which is initialized with a set of n random MAC addresses. This initialization aligns with the previously introduced *anonymization noise* technique [3]. Building upon the definition of $\gamma-$deniability in [4], we introduced the *anonymization noise* to control the value of γ and make it equal to 1. This establishes the formal concept of 1-deniability, ensuring that for every MAC address present in the Bloom filter, there exists at least one other MAC address not in the filter that, if queried, would yield an indistinguishable result from the original, thereby allowing us to deny its presence with certainty. Essentially, for every element stored, there is at least one other element not in the Bloom filter that, if added, would not change the filter's bitmap. With the added security of the 1-deniability property, we can securely transmit the Bloom filters over the network and perform server-side intersections. This allows us to analyze the flow of people within a specific time window or simply count the number of elements stored within a Bloom filter.

The remainder of this paper is structured as follows: In Section II, we delve into pertinent related work. Section III provides an in-depth exploration of the methodology and techniques employed in designing our privacy-preserving crowd

monitoring solution. Subsequently, Section VI presents an in depth analysis on the anonymization noise applied to Bloom filters, showcasing the effectiveness of 1-deniability and introducing the concept of $\gamma - K-$ anonymity. Section VII unveils the flows-monitoring architecture, offering a comprehensive overview of the entire pipeline, starting from data detection and culminating with Bloom filter intersection for harnessing flow data. Lastly, in Section VIII, we present our concluding remarks and insights.

II. Related Works

In recent years, the importance of estimating crowd sizes and understanding people's movements in specific areas has become increasingly evident. This knowledge can significantly improve the overall management of various situations and events.

Over the past few years, numerous techniques have been proposed to address the challenge of people counting. In [5], [6], the authors exploit a multi-camera surveillance system. Their method combines partial body detection and person re-identification to accurately count individuals in overlapping areas. In contrast, recent works, such as [7], [8], have employed LiDAR sensors, which, compared to video camera techniques, address privacy issues. However, in both solutions, hardware costs play a pivotal role and the suitability of LiDAR in different environmental scenarios remain problematic. Moreover, due to the recent regulations imposed by the GDPR [1], recording and storing face detection data raises privacy concerns.

Conversely, studies presented in [2] and [9] explore the use of WiFi probe request messages as an effective method for crowd-monitoring in various scenarios, achieved by collecting WiFi fingerprints from mobile devices. Solutions utilizing such messages can be applied in both indoor and outdoor settings, demand budget-friendly equipment, have low computational requirements, and can mitigate privacy concerns. New approaches investigate the possibility to apply artificial intelligence algorithms to divide probe requests to identify the single devices that sent them. An example is explained in [10] and [11], where the authors employed clustering techniques to retrieve groups of messages, each one representing a device. The former focused on some probe requests fields, such as MAC addresses, arrival timestamp, throughput capabilities and received signal strength indicator (RSSI). While the latter considered the length of probe request fields values. Nevertheless, WiFi fingerprints have their limitations, primarily linked to the absence of ground truth data essential for testing and fine-tuning counting algorithms. In particular, all the reported works obtained some results that are not compared with any certain ground truth. Indeed, when analyzing collected WiFi fingerprints, knowing the real number of active devices can only be performed through expensive methods, such as manually counting the people presence. This process is challenging in densely populated environments.

In light of recent developments in European regulations, the GDPR [1] imposes significant restrictions on the storage and management of sensitive information. As previously mentioned, mobile devices regularly emit probe request messages that contain details, such as MAC addresses, which are crucial for device identification and monitoring. Notably, the GDPR categorizes MAC addresses as personal data, necessitating the implementation of privacy protection measures [12]. Numerous approaches have been presented in literature to tackle this challenge, including [13] and [14]. Both these solutions address privacy concerns by utilizing Bloom filters to store MAC addresses information and employing an asymmetric homomorphic encryption system to process the data within the Bloom filter. Unfortunately, as demonstrated in [4], while Bloom filters can indeed safeguard the privacy, they may fail to meet the stringent anonymity constraints mandated by the GDPR. Additionally, the authors of [4] introduce two essential concepts for safeguarding anonymity: $\gamma-$ deniability and $\gamma - K-$ anonymity. In simple terms, a small number of inserted elements do not guarantee anonymity. Regrettably, this observation is not taken into account in [13] and [14], rendering their proposed solutions viable only for sufficiently large crowds.

In this current work, we present an extensive analysis of privacy techniques applicable to Bloom filters. These techniques are designed to meet the stringent privacy requirements mandated by GDPR regulations. Additionally, we introduce a crowd-monitoring framework, which is focused on understanding the dynamics of crowd movement across multiple scanners. Importantly, all of our approaches adhere to GDPR-compliant privacy standards, guaranteeing the protection of user data.

III. Crowd monitoring through WiFi Probe Request messages

As we have detailed earlier, our methodology harnesses WiFi signal detection and subsequent data analysis to precisely estimate the presence and variety of smart devices, encompassing smartphones, tablets, laptops, and smartwatches, among others. Our method involves scanning the WiFi spectrum to capture packets emitted by these smart devices. The process begins when a smart device activates its WiFi interface, emitting probe requests as it searches for nearby Access Points (APs) – a fundamental step for establishing a connection to a WiFi network. In the ensuing discussion, we will provide an overview of the fundamental characteristics of WiFi that we exploit to garner valuable insights from the probe request messages. Furthermore, we will introduce the Bloom filter data structure, which is integral to our development of the concept of 1-deniability.

Probe requests constitute a specific category of WiFi management frames primarily utilized for the purpose of network discovery. Notably, these frames lack encryption, as their primary role is to facilitate the identification of available networks. Whenever a device activates its WiFi interface, it automatically initiates the transmission of these broadcast messages. Furthermore, even when a device is already connected to a network, it continues to send probe request messages with the objective of identifying potentially better access points, thereby aiming to enhance the quality of its network connection. Upon dispatching a probe request, a Probe Timer is initialized. If the device fails to receive a response, it

Fig. 1: Flows-monitoring pipeline through WiFi scanner leveraging WiFi probe request messages.

automatically switches to the next channel frequency and reiterates the network discovery process. In the event a probe response is received, the device proceeds to initiate the authentication process. This authentication procedure involves the exchange of authentication frames with the access point, ensuring validation of the device's adherence to IEEE 802.11 standards and compatibility with the target network.

It is worth noting that this behavior bears a resemblance to Bluetooth, where similar messages are referred to as *inquiry requests* and *inquiry responses*. These Bluetooth messages serve a parallel purpose of discovering nearby devices and establishing connections.

Within the probe request messages, the MAC address serves as a primary key field. This 48-bit identifier is designed to uniquely distinguish each device globally. However, since 2014, device manufacturers have increasingly adopted MAC address randomization techniques as a means of bolstering user privacy [15]. These privacy-enhancing techniques involve the use of "pseudo" or "fake" MAC addresses when transmitting probe request messages. The objective of this technique, denoted as *MAC randomization*, is to make it more challenging to track a device, thereby safeguarding people's identity. Notably, there is no standardized method for randomizing MAC addresses, resulting in a variety of methods employed by different vendors.

In our earlier research [16], we highlighted this evolving trend, particularly prevalent in newer devices. Our observations have revealed variations in MAC address randomization practices among these devices. While some devices choose to randomize the entire 48-bit MAC address, others selectively randomize only the second half of the address, retaining the first 24 bits, which are known as the Organizationally Unique Identifier (OUI). This selective approach to randomization strikes a balance between user privacy and network compatibility, ensuring seamless interaction with existing network infrastructure while enhancing personal data protection.

Some past works focused on derandomization methods, which are able to identify approximately the same device behind a set of random MAC addresses. Works [2], [10], [11], [17] represent only a portion of the extensive body of literature contributing to the field of derandomization techniques in the context of probe request messages. Actually, such methods are heuristic and their accuracy depends on the device models and on the usage conditions.

A possible framework designed for the analysis of crowd monitoring and the study of people's movements, based on multiple access points distributed over a wide area, is depicted in Figure 1. It provides an overview of the operations conducted by the sniffer. The process starts with the sniffing of the WiFi probe request messages, followed by the application of a filter, i.e., based on RSSI signal strength. This filter is instrumental in eliminating potential outliers from the analysis. Subsequently, a derandomizer script is employed to counteract the effects of MAC address randomization. Finally, the resulting MAC addresses are stored in a Bloom filter, which is initially populated with random MAC addresses, referred to as *anonymization noise*.

IV. Bloom filters as a privacy tool

Bloom filters are a probabilistic data structure well known in the literature [18], [19] which have been introduced to solve the approximated set membership problem. They have been devised to optimize the performance of data storage systems, whenever a set must be efficiently implemented with minimum memory footprint.

A Bloom Filter (BF) is used to represent a collection of elements. It is constructed using an array of bits, represented as $BF \in \{0,1\}^m$, where m is the array's length, and k independent hash functions, denoted as $H_1, H_2, \ldots, H_k$. These hash functions map an input element x to one of the m bits within the bit array. We refer to the i-th bit of BF as $BF[i]$. Initially, all bits are initialized to 0. When adding an element x to the Bloom filter (a visual representation can be found in Figure 2), the k hash functions are applied to x, and the bits in BF associated with the positions generated by the hash functions are set to 1:

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

To confirm the presence of an element in a Bloom filter, the element undergoes the same set of k hash functions, and the resulting output is cross-referenced with the current values of the corresponding bits in the Bloom filter. If all the 1s in the output align with the corresponding bits in BF (i.e., both are set to 1), the element is regarded as *likely present* in the Bloom

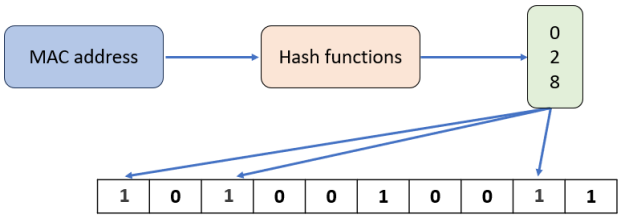

Fig. 2: When adding a new element to the Bloom filter, the input is processed using a set of k independent hash functions. The resulting values from these hash functions (i.e., 0, 2, 8) serve as indices to access the Bloom filter array. At each corresponding index, the bit is set to 1. This process is repeated for each new input value.

filter. This means that an element could be considered within the Bloom filter, even if this is not correct. This event is named "false positive". On the contrary, if even a single bit in the match is set to 0, the element is conclusively deemed absent in the Bloom filter. It is crucial to emphasize that a Bloom filter has the potential to yield false positives, i.e., it may incorrectly indicate that an element is present in the set when it is not. However, it does not yield false negatives, meaning it cannot erroneously indicate that an element is absent when it is, in fact, present.

Notably, it is only possible to add elements in a Bloom filter, and it is not possible to remove elements. Thus, for the WiFi scanner scenario, the Bloom filter keeps accumulating MAC addresses and its probability of false positive increases, until it saturates (i.e., all bits are ones). To avoid saturation, the bloom filter must be periodically reset.

It is worth highlighting that:

- A smaller value of k raises the number of 0 bits in the array, making them more likely to be available for elements that are not part of the set S of elements stored in the Bloom filter.
- Conversely, a larger value of k heightens the likelihood of encountering at least one 0 bit for an element that is not a member of S.

To minimize false positives, using standard results [19], it is possible to determine analytically the optimal value of k based on the available memory m according to (2):

$$k_{opt} = \frac{m}{n} \log(2) \tag{2}$$

Bloom filters offer a versatile solution for storing MAC addresses observed by a WiFi scanner. To illustrate the process of parameter tuning, we will examine three different scenarios where the Bloom filter is reset periodically with a period denoted as T.

In the first scenario, denoted as A, at most 1,000 people are expected to be observed by the scanner in a period of time of 120 s, so the flow rate is approximately 8 people per second. Assuming to have a memory of 10,000 bits, the optimal value k_{opt} of hash functions, according to (2), is equal to 7. This first scenario can be a use case where we want to monitor the flow of people in a very crowded plaza. Let us now consider a different scenario, denoted with B, where we want to analyze the flow of people and cars at an intersection during the green light phase. In this case, the period of time is shorter than before, let us assume it is 40 s and that we expect to collect 4 probe requests per second, with only 1,000 bits available. In this case the value of k_{opt} is 4. Finally, we will focus on a different use case, e.g., the monitoring of people entering a lecture hall in the morning. In this case, we can use a longer period, e.g., 10 minutes, with an expected turnout of around 200 people. With a 2,000-bit Bloom filter, the optimal value for k would be 8. Table I provides a concise summary of the three previously mentioned scenarios.

Tuning the parameters of a Bloom filter is a fundamental aspect related to the specific use case under consideration, because it depends on the number of bits available, the length of the detecting time window and the number of people/devices expected to be detected.

To determine the count of elements stored within a Bloom filter, following [20], we can resort to the following formulation (3):

$$c = -\frac{m}{k} \log\left(1 - \frac{t}{m}\right) \tag{3}$$

where m corresponds to the total number of bits in the Bloom filter, k represents the count of used hash functions, and t indicates the number of bits set to 1.

V. Deniability and Anonimity

False positive events have been traditionally considered as weaknesses, but authors in [4] instead showed the potential of using Bloom filters to "hide" the presence of elements in the bitmap, thus achieving formal levels of privacy. Specifically, they introduced the concepts of deniability and anonymity.

Deniability refers to the ability to plausibly deny the presence or association of specific elements within the Bloom filter. It implies that, even if an entity gets hold of the Bloom filter, there can be no certainty as to whether a particular element was added or not.

Anonymity, instead, pertains to the protection of the identity or specific information associated with the elements stored in the Bloom filter. It ensures that the elements themselves remain concealed or unidentifiable when queried or retrieved from the Bloom filter.

In a more formal way, let $BF(S)$ be a Bloom filter storing a set S of elements:

Definition 1: (taken from [4]) **Hiding Set**: A set V is called *Hiding Set* for a Bloom filter $BF(S)$ if V contains all the elements $v_i \in U$ such that $v_i \notin S$ and a query for v_i in the Bloom filter returns 1. Where $|U|$ represents a large set, approximately equal to $2^{48} \approx 2.8\ 10^{14}$.
In other words, a hiding set is a set of elements not present in the Bloom filter that, when queried, falsely indicate their presence in the filter.

Figure 3 shows an example of "Hiding Set". In this case, a 10-bit Bloom filter, using 2 hash functions, has been employed to insert 3 elements x_1, x_2, x_3. Additionally, 3 elements v_1, v_2, v_3, belonging to the Hiding Set are incorrectly identified as false positives when querying the filter, yielding positive results for these elements.

Definition 2: (taken from [4]) $\gamma-$ **Deniability**: An element $x \in S$ inserted in $BF(S)$ is defined *deniable* if $\forall i \in \{1..k\}$ exist at least one element $v \in V$, such that $\exists j \in \{1..k\}$ such that $H_i(x) = H_j(v)$. A $BF(S)$ is $\gamma-$ *deniable* whenever a randomly chosen element $x \in S$ is deniable with probability γ.

In other words, an element is termed "deniable" when it can be replaced with items not originally included in the Bloom filter's stored set, all without altering the filter's bit map.

TABLE I: Real-world examples for Bloom filter settings

Scenario	person/s	T	m	k_{opt}
A	8	120 s	10,000 bit	7
B	4	40 s	1,000 bit	4
C	0.3	600 s	2,000 bit	8

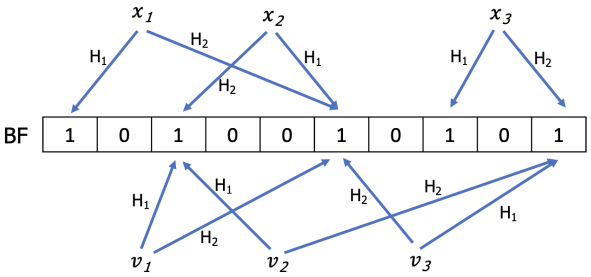

Fig. 3: Elements v_1, v_2, v_3 belonging to the hiding set of Bloom filter of 10 bits storing x_1, x_2, x_3. Arrows indicate the bits that are set to one according to two hash functions H_1 and H_2.

It is essential to emphasize that the 1-deniability approach ensures that an element correctly inserted into the filter is indistinguishable from just one element that was never part of the filter to begin with.

To enhance the level of protection, we can resort to the concept of anonymity outlined in Section 3.

Definition 3: (taken from [4]) $\gamma - K-$**Anonymity**: Considering a Bloom Filter $BF(S)$ and $x \in S$ inserted in $BF(S)$, x is $K-$ *Anonymous* if exists at least $K-1$ hiding set elements $\langle v_1 ... v_{K-1} \rangle \in V$, with $K \geq 2$, such that $\forall i \in \{1 \dots k\}$ $\exists \langle j_1 ... j_{K-1} \rangle \in 1..k$ such that $H_i(x) = H_{j_1}(v_1) = ... = H_{j_{K-1}}(v_{K-1})$. Consequently, it is possible to say that a Bloom filter $BF(S)$ is $\gamma - K-$ *anonymous* if each randomly chosen element is $K-$ anonymous with probability γ.

Using (4) proposed in [4] is possible to compute the level γ related to the $\gamma - K-$ anonymity property for a specific Bloom filter storing n MAC addresses, according to the following formula, for $K \geq 2$:

$$\gamma(K, BF(S)) \approx \left(1 - \exp\left(-\frac{hk}{m(1-e^{-kn/m})}\right) \times \sum_{i=0}^{K-2} \frac{\left(\frac{hk}{m(1-e^{-kn/m})}\right)^i}{i!}\right)^k \quad (4)$$

with h computed as

$$h = (|U| - n)(1 - e^{-kn/m})^k \quad (5)$$

where $|U|$ is the number of all the possible MAC addresses, equal to 2^{48}.

VI. Anonymized Counting

We introduce the concept of *anonymization noise* applied to a Bloom filter. It is a privacy-enhancing technique that introduces uncertainty and randomness into the Bloom filter's data, making it more challenging to infer specific elements from the filter's contents. This concept is particularly useful when preserving the anonymity and confidentiality of data elements stored in a Bloom filter is of utmost importance.

The anonymization noise involves adding random mac addresses to the Bloom filter whenever the Bloom filter is reset. These random mac addresses are not associated with any actual data but are used to obscure the data subsequently stored in the filter. The inclusion of anonymization noise makes it impossible, if the 1-deniability property is satisfied, for an external observer to make any assumption if an element is stored in the Bloom filter or not, thereby protecting the anonymity of the actual elements. This adds a layer of privacy and security to the Bloom filter. It is worth to note that while anonymization noise enhances privacy, it can also introduce a trade-off by potentially increasing the rate of false positives. If the anonymization noise introduced is too much then we will have a very good privacy but a bad accuracy as the false positive rates increase. While, too little noise will brings to bad privacy, not satisfying the 1-deniability property, but a good accuracy in terms of false positive rate.

In essence, it is essential to strike the right balance when determining the optimal number of fake elements to be inserted as anonymization noise. This balance ensures that the level of privacy and anonymity is maximized while minimizing the impact on the filter's performance and accuracy.

Due to the recent GDPR regulations in terms of users' private information, sniffing and analyzing WiFi probe request messages has become a sensitive issue. Therefore, it is crucial to incorporate *anonymization noise*. This ensures that from the very first insertion of legitimate MAC addresses, each of them is shielded by a minimum of $K-1$ additional elements.

Figure 4 visually represents the results of applying (4) to a Bloom filter with $m = 10,000$ bits and $k = 7$, for various values of K. This graph facilitates the evaluation of $c_{\min}^K$ in a manner that ensures, for each element randomly drawn from the filter, there are at least $K-1$ non-inserted elements available to provide the necessary cover and privacy protection. E.g., setting $c_{\min} = 30$ is enough to guarantee $1 - K-$ anonymity for $K = 2, 3, 4$.

We now explore the counting process in a Bloom filter while ensuring the 1-deniability through the anonymization noise. In Algorithm 1, we present the pseudocode to demonstrate

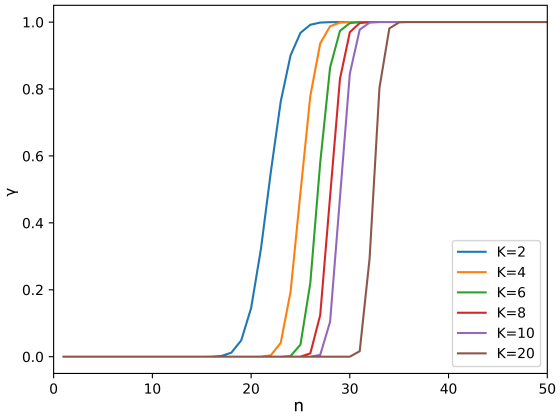

Fig. 4: Achievable level of γ-deniability, for different values of anonymity K, when n MAC addresses are inserted into a Bloom filter with $m = 10,000$ bits and $k = 7$.

the key steps involved, including initialization of the Bloom filter, insertion, and count estimation while accounting for the anonymization noise.

More in detail, at the beginning of every new iteration the Bloom filter is initialized with all bits equal to 0. Then the random noise, consisting in $c_{\min}$ "fake" MAC addresses, is inserted into the Bloom filter. Notably, thanks to the standard properties of independent hash functions used in Bloom filters, an efficient implementation of this step requires just to generate $c_{\min} \times k$ random positions in the Bloom filters, since this is equivalent to adding $c_{\min}$ elements with k hash functions. After this initialization phase, for every new probe request detected, the MAC address is inserted into the filter, setting the proper bits to 1. The indexes of the bits are given by the output of the k hash functions applied to the MAC address. When the capturing window is over, the first operation is counting the number of ones in the Bloom filter and then (3) is applied to estimate the number of element present in the filter. Finally from the estimated value, a $c_{\min}$ value is subtracted, due to the anonymization noise inserted at the beginning.

Algorithm 1 Counting algorithm with anonymization noise

Input: Bloom filter (BF) of size m bit, Number of elements $c_{\min}$ for the anonymization noise, k independent hash functions
Output: Estimated count of elements inserted

procedure RESET($BF, c_{\min}$)
  for $i = 1, \ldots, m$ **do**
    $BF[i] = 0$ ▷ Reset each bit
  end for ▷ Add the anomization noise
  for $c = 1, \ldots, c_{\min} \times k$ **do**
    $i =$**random-int**$(1, m)$ ▷ Choose a random bit to set
    $BF[i] = 1$ ▷ Update the bit
  end for
end procedure

procedure INSERT(BF, mac)
  for $i = 1, \ldots, k$ **do**
    $BF[H_i(\text{mac})] = 1$ ▷ Set bit to 1 in the H_i index
  end for
end procedure

procedure COUNT($c_{\min}$)
  $t = 0$ ▷ Init t
  for $i = 1, \ldots, m$ **do**
    if $BF[i] = 1$ **then**
      $t = t + 1$ ▷ Count number of 1 in BF
    end if
  end for
  $c = -\frac{m}{k} \log\left(1 - \frac{t}{m}\right)$ ▷ Apply (3)
  return $c - c_{min}$ ▷ Compensate for the anonymization noise
end procedure

VII. Application to Crowd-flow analysis

Analyzing crowd flows through WiFi probe requests entails identifying common MAC addresses among various WiFi scanners. Consequently, the analysis cannot be performed within a single scanner; instead, the data from each scanner must be transmitted to a central server. This central server then analyzes the data from different scanners to derive insights and valuable information regarding crowd flows. In order to preserve privacy in this task, we exploit Bloom filters beyond the representation of individual sets; indeed, they can also facilitate set unions and intersections. The intersection operation is particularly crucial, especially in the context of flow detection. The core concept is that by performing the intersection of two Bloom filters, we can determine the number of MAC addresses that have been detected in both Bloom filters. Consider two subsets, $S1$ and $S2$, derived from the universal set U. Each subset is respectively represented by Bloom filters $BF_1(S1)$ and $BF_2(S2)$, configured with the same parameters (m and n). In order to find the intersection between these subsets, a bitwise logical AND operation is executed between the two Bloom filters. This operation yields a new Bloom filter:

$$BF_3 = BF_1 \wedge BF_2$$

representing the intersection. To determine the count of elements within this intersection, taking into account the anonymization noise, we can leverage (6), derived from [21].

$$c = \frac{\log\left(m - \frac{t_3 \times m - t_1 \times t_2}{m - t_1 - t_2 + t_3}\right) - \log(m)}{k \times \log(1 - \frac{1}{m})} - c_{\min} \quad (6)$$

where t_i represents the number of bits set to 1 in BF_i. This approach allows for efficient determination of the cardinality of the intersection between the sets represented by these Bloom filters.

It is worth to notice that the approach can be extended to the intersection of any number of Bloom filters, allowing to identify very specific paths across a sequence of WiFi scanners.

A. Numerical evaluation

In order to evaluate the accuracy of the intersection between two Bloom filters we developed a simulator in Python, where we started from two empty Bloom filters, namely BF_1 and BF_2, with the configuration of $m = 10,000$ and $k = 7$. We set $c_{\min} = 30$, according to the reasoning done when discussing Figure 4. Then we proceeded through the following steps:

1) Reset each Bloom filter with anonymization noise equal to $c_{\min}$.
2) Insert 200 random MAC addresses into BF_1.
3) Insert 200 random MAC addresses into BF_2.
4) Generate n_c random MAC addresses and insert all of them into both BF_1 and BF_2.
5) Compute the resulting Bloom filter BF_3 computed from the intersection of BF_1 and BF_2.
6) Count the MAC addresses observed in both WiFi scanners based on (6) (the subtraction of $c_{\min}$ is employed to compensate the anonymization noise).

At the end of step 3, by construction, the two Bloom filters have no common MAC addresses. At the end of step 4, the two Bloom filters have stored $200 + c_{\min}$ different MAC addresses, plus n_c common MAC addresses, modeling n_c devices detected by both scanners.

Table II shows the estimated number of common MAC addresses, computed on BF_3, and compares it to the actual

TABLE II: Numerical results of the estimation of common MAC addresses

Common MAC addresses (real n_c)	Common MAC addresses (estimated n_c)
0	0.31
10	10.02
50	50.99
100	100.54
200	200.76
500	500.58

number n_c. The results show an accurate estimation of the common MAC addresses. Notably, when no common address is present in the two Bloom filters (i.e., $n_c = 0$), the estimator is still quite accurate, even if $200 + c_{\min}$ distinct MAC addresses were previously inserted for each of the Bloom filters. This is due to the approximations behind the adopted formulas and the adopted probabilistic approach.

Figure 5 shows the relative error in estimating the common MAC addresses, based on the complete data set from which the results in Table II were derived. The relative error is evaluated through (7).

$$E_r = \frac{|\text{real } n_c - \text{estimated } n_c|}{\text{real } n_c} \tag{7}$$

Of course, the relative error remains very small even for a large number of common MAC addresses (approximately 500). This is attributed to the optimization of the Bloom filter, which has a size of $10,000$ bits and utilizes 7 hash functions. The design is tailored to minimize the false positive ratio effectively, even when inserting $1,000$ elements into the filter.

B. Implementation

As illustrated in Figure 1, the comprehensive data capture and processing pipeline terminates with the insertion of detected MAC addresses into a Bloom filter. This Bloom filter is subsequently relayed over the network to a central server, which serves as the hub for collecting, storing, and

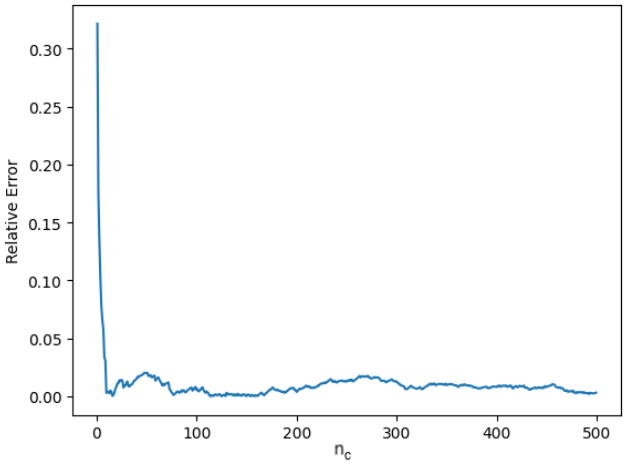

Fig. 5: Relative error E_r in counting the number of MAC addresses resulting from the intersection of two Bloom filters.

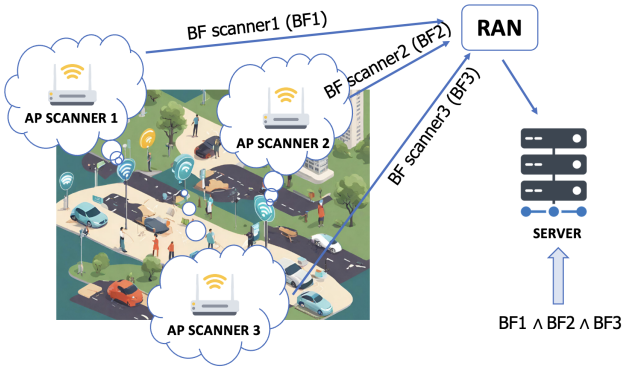

Fig. 6: Architecture for crowd-flow analysis. Each AP scanner sends periodically its Bloom filter to the server, where all the Bloom filters are processed to infer crowd flows.

processing incoming Bloom filters from various access points. The architectural depiction in Figure 6 provides a visual representation of this seamless flow.

More in detail, we employed a Raspberry Pi 4B as an Access Point (AP) scanner, utilizing an external WiFi dongle and `tshark` [22] software to capture probe request messages from nearby devices. Subsequently, a Python script executes the workflow outlined in Figure 1, generating as output the Bloom filter. All the Bloom filters from all the AP scanners are saved in the server, where a SQL database is employed to store the collected data.

This approach extends our capabilities beyond merely assessing individual Bloom filters derived from single scanners. It allows us to perform intersections on the Bloom filters originating from diverse access points, each corresponding to distinct geographical areas. Through this intersection process, we gain the capacity to extract precious insights into the movement and flow of individuals across these areas. This level of analysis yields rich data that is instrumental in understanding crowd dynamics and behavior.

VIII. Conclusions

In this paper we have tackled the significant challenge of crowd counting and tracking within large-scale gatherings, with a strong focus on the growing importance of privacy and compliance with regulations, such as the European GDPR.

Our proposed crowd monitoring framework leverages IoT technology and utilizes WiFi probe request messages as a key tool to offer an innovative solution that not only accurately characterizes people flows and quantifies crowd size but also places paramount importance on preserving individual privacy. The core objective is to ensure precise people counting while simultaneously upholding the principles of privacy, aligning our approach with the stringent regulations set out in the GDPR. We have employed advanced techniques like 1-deniability and anonymization noise within Bloom filters to guarantee the formal deniability of each element's presence. This approach ensures privacy when transmitting data over the network and conducting server-side intersections for flow analysis.

Looking ahead, our future research endeavors will focus on further refining these techniques and exploring additional privacy-enhancing concepts to advance the field of crowd counting and monitoring. Additionally, we are planning to develop an advanced machine learning-driven derandomization algorithm to enhance device counting accuracy within the coverage of an AP scanner. Furthermore, within the European project TrialsNet [23], we plan to deploy several AP scanners in a public area to rigorously test the effectiveness of our solution.

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] M. Nitti, F. Pinna, L. Pintor, V. Pilloni, and B. Barabino, "iABACUS: A wi-fi-based automatic bus passenger counting system," *Energies*, vol. 13, no. 6, p. 1446, 2020.

[3] R. Rusca, A. Carluccio, D. Gasco, and P. Giaccone, "Privacy-aware crowd monitoring and wifi traffic emulation for effective crisis management," in *2023 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, 2023, pp. 1–6.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] A. Simončič, M. Mohorčič, M. Mohorčič, and A. Hrovat, "Non-intrusive privacy-preserving approach for presence monitoring based on WiFi probe requests," *Sensors*, vol. 23, no. 5, 2023.

[11] M. Uras, E. Ferrara, R. Cossu, A. Liotta, and L. Atzori, "MAC address de-randomization for WiFi device counting: Combining temporal- and content-based fingerprints," *Computer Networks*, vol. 218, p. 109393, 2022.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 131–155, 2012.

[19] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004. [Online]. Available: https://doi.org/10.1080/15427951.2004.10129096

[20] S. J. Swamidass and P. Baldi, "Mathematical correction for fingerprint similarity measures to improve chemical retrieval," *Journal of chemical information and modeling*, vol. 47, no. 3, pp. 952–964, 2007.

[21] 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.

[22] Tshark. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[23] Trialsnet EU project. [Online]. Available: https://trialsnet.eu/

Riccardo Rusca received the Dr.Ing. degree in computer science engineering from the Politecnico di Torino, Italy, in 2020. He is a PhD student in the Department of Control and Control Engineering, Politecnico di Torino. His main area of interest is crowd monitoring and city sensing techniques supported by next generation mobile networks.

Alex Carluccio received the Dr.Ing. degree in computer science engineering from the Politecnico di Torino, Italy, in 2023. His main area of interest is the design of privacy-preserving methods for crowd monitoring.

Claudio Casetti is a Full Professor at the Department of Control and Computer Engineering, Politecnico di Torino, Italy. He has published over 250 papers in peer-refereed international journals and conferences on the following topics: vehicular networks, Intelligent Transportation Systems, 5G/6G networks, IoT systems. According to Google Scholar, his H-index is 42. He is a Senior Member of IEEE and a Senior Editor for Mobile Radio of IEEE Vehicular Technology Magazine.

Paolo Giaccone received the Dr.Ing. and Ph.D. degrees in telecommunications engineering from the Politecnico di Torino, Italy, in 1998 and 2001, respectively. He is a Full Professor in the Department of Electronics, Politecnico di Torino. His main area of interest is the design of optimal control algorithms and of resource allocation algorithms in next generation networks.