Stop watching me! Moving from data protection to privacy preservation in crowd monitoring

Fatemeh Marzani, Thijs van Ede, Geert Heijenk, and Maarten van Steen

University of Twente, The Netherlands
{f.marzani, t.s.vanede, geert.heijenk, m.r.vansteen}@utwente.nl

Abstract. The monitoring of large crowds is essential to optimize traffic flows, ensure safety at large-scale events, and plan effective evacuation routes during emergencies. However, such monitoring rightfully leads to privacy concerns, especially when tracking individuals rather than groups. Existing approaches attempt to address these concerns by pseudonymizing personally identifiable information and restricting the analysis to statistical counts. However, these methods fail to preserve privacy, particularly when small groups can be correlated with external data. To combat this issue, we leverage the idea that crowd monitoring applications are interested in only large crowds (e.g., > 100 people) and can deal with low noise levels (e.g., it does not matter whether we count 95 or 105 people). We propose and evaluate two methods that not only protect individual data, but also enhance privacy by introducing varying levels of controlled noise: higher for smaller groups and lower for larger crowd movements. These methods include probabilistically: (1) sampling hash functions and (2) sampling detected identifiers. We show that our methods significantly reduce the risk of re-identification in small crowds while maintaining high precision in large crowd estimations, making them highly effective for privacy-preserving crowd monitoring.

Keywords: Crowd monitoring, pedestrian dynamics, privacy preservation, Bloom filters, homomorphic encryption, privacy-by-design.

1 Introduction

Comprehending the crowd dynamics has been a long-standing focus in scientific research. Data collected from crowd behavior can be applied in various fields such as urban planning [1], tourism [2], and enhancing safety and security [3]. Automatic measurement of crowd dynamics enables the gathering of more precise data and is more convenient compared to manual methods. To automate the process, scanners can be installed in public spaces to collect unique identifiers, such as MAC addresses of mobile devices or public transport card identifiers, for each individual. By gathering these identifiers (which can be subject to erroneous detections), interested parties can estimate the crowd size near those scanners, as well as the size of the flows between them. Handling data related to crowds has always been a sensitive and complex challenge. In this work, we distinguish two situations. The first is to count individuals at a specific location and time. The second and most challenging situation involves counting people over time (and perhaps across multiple locations). [1]

In the first case, we can resort to counting detections of unique identifiers during a very short measurement interval (say, a few seconds to at most a few minutes), after which identifiers can be discarded. This limits storing identifiers to a location and time in which they were collected and leaves only the count for further processing.

However, in the second case, we need to store identifiers for future re-identification. Even if an identifier itself cannot be used for the identification of a natural person, as is the case with pseudonyms, storing an identifier for re-identification can easily lead to recognition of patterns that *do* lead to such identification. For example, Montjoye et al. [4] discuss that just four spatio-temporal data points are sufficient to uniquely identify 95% of individuals.

In the context of crowd monitoring systems, there are two fundamental issues that must be considered to prevent individuals from being tracked. First, data protection, which refers to ensuring that identifiers and even pseudonyms are not disclosed to unauthorized parties. Second, ensuring data privacy, which refers to preventing (at all times) the identification of natural persons to unauthorized parties when given the information provided by that system. Although there is existing literature on data protection, ensuring data privacy is much harder, as we cannot foresee which additional information may be available to a party using the system or how the provided information will be used. We observed that existing methods, at best, provide data-protection techniques to count individuals at a specific time. Some approaches also provide data protection over time using anonymous identifiers or encrypted data [5], yet fail to protect privacy. Our novel contribution is to introduce a method that ensures both privacy and data protection.

In our approach, we trust the detecting devices but do not extend this trust to any other party, including the server responsible for performing further processing. For data protection, we store the pseudonyms in Bloom filters (BFs), i.e. probabilistic data structures supporting approximate set operations. After encryption and shuffling, these Bloom filters can be used only for intersection operations and cardinality estimations (and not for membership testing). Encryption ensures that even if the server is compromised, it cannot extract individual identifiers from the data. Shuffling further enhances protection by randomizing the order of bits in the BF, preventing an attacker from inferring membership from the bit positions. Given our data protection measures, breaching privacy is still possible when only a few people are counted: if we count only 1 person at location A, then having additional information about occupancy at A, re-identification can be relatively easy. To prevent such re-identification attacks, we intentionally introduce uncertainty *especially* when counting a small number of individuals, yet retain high precision for large groups. The intuition behind our approach

[1] In this paper, we use pedestrian monitoring as an example, but we can also count more diverse groups of people.

is that, in small groups, sampling may not provide a precise representation of the population, which can help protect privacy. However, in larger groups, the sample size also increases, resulting in more precise and reliable estimations. We use this property to deliberately reduce the precision of small crowd estimates in favor of privacy protection. We discuss two different sampling methods:

1. Sampling hash functions
2. Sampling identifiers

The source code for evaluating our methods is publicly available.[2]

2 Related Work

For years, monitoring crowd behavior has been a focus of research, taking advantage of technologies such as Wi-Fi signals and unique device identifiers (e.g., MAC addresses) to estimate crowd densities, flows, and mobility patterns [6,7,8]. These systems typically rely on the detection of signals emitted by devices carried by individuals, such as smartphones or other Wi-Fi-enabled devices. By capturing and analyzing these signals, researchers can derive valuable insights about crowd behavior. However, these methods often compromise privacy by allowing the tracking and profiling of individuals without their consent [9].

To address these concerns, randomization of MAC addresses was introduced as a countermeasure. This approach ensures that devices periodically generate and use fake MAC addresses instead of their real ones. Although this method mitigates tracking to some extent, studies have shown that inconsistent implementations between different manufacturers still allow reidentification [10]. This limitation exposes individuals to potential privacy breaches. Moreover, for our use case, this limitation significantly impacts the accuracy of crowd flow estimation, undermining the reliability of the data collected for monitoring purposes.

Alternative approaches, such as pseudonymization using hash functions or encryption, have also been explored to mask identifiers [11,12]. In pseudonymization, original identifiers are transformed into pseudonyms through methods such as one-way hash functions or deterministic encryption schemes. However, due to the limited identifier space (e.g., MAC addresses are effectively limited to 2^{24} bits), pseudonyms remain vulnerable to brute-force attacks [13,14]. These weaknesses highlight the need for stronger privacy-preserving mechanisms.

Some methods focus on aggregating data to enhance privacy. Linear counting sketches [15] and k-anonymity-based approaches [11] enable crowd size estimations by grouping data together, making individual identifiers less distinguishable. Similarly, encrypting identifiers in Bloom filters ensures that statistical counts can be obtained without exposing individual identities [5]. However, these methods are not foolproof, as linkage with external data sources can lead to reidentification, especially in sparsely populated areas.

Other privacy-preserving systems, such as DEVCNT [16], avoid completely collecting unique identifiers. Instead, DEVCNT estimates the number of devices

[2] `https://anonymous.4open.science/r/private-bloom-filter/`

in a crowd by detecting and counting active scan events. Although this avoids direct data protection issues, it limits the system to simple counting operations and lacks support for more complex queries, such as intersections or crowd-flow size estimation across locations. These methods collectively underscore the persistent challenge of balancing utility with robust privacy. Although existing techniques offer partial solutions, they often fail to provide the comprehensive privacy guarantees required for practical crowd monitoring systems.

RAPPOR [17] employs a different strategy using differential privacy. It encodes data into Bloom filters and introduces random noise to obscure individual contributions. RAPPOR is specifically designed for web tracking applications, such as extracting the most frequently visited websites, and does not address Bloom filter intersection challenges for computing the size of crowd flows. Ke et al. [18] introduced DPBloomfilter, which integrates the Randomized Response mechanism into the Bloom filter to achieve differential privacy for membership queries. Their approach addresses the risks of individual data leakage in standard Bloom filters. Their work primarily focuses on static membership queries, not addressing dynamic crowd flow estimation across locations or epochs, a key feature of our system. Rusca et al. [19] proposed a WiFi-based crowd monitoring system using Bloom filters with fixed initial noise to ensure deniability. However, this added noise cancels out during Bloom filter intersections, making their approach unsuitable for our goal, preserving uncertainty in small group counts during crowd flow analysis.

Recent research has also focused on developing location privacy-preserving techniques for location-based services (LBSs), primarily to prevent adversaries from tracking users, inferring movement patterns, and profiling them. As surveyed by Jiang et al. [20], common approaches include spatial cloaking, dummy locations, differential privacy, and cryptographic techniques such as homomorphic encryption and secure multiparty computation. These methods typically obfuscate location data to protect individual privacy. However, our focus differs from these approaches as we aim to develop a crowd-counting mechanism that preserves privacy. Our method ensures that when the number of people in a location is small, the exact count remains uncertain by design to prevent re-identification. Since LBS privacy techniques focus primarily on anonymizing movement data, they are not directly applicable to our problem.

3 System and Threat Model

Figure 1 shows the setup of our crowd monitoring scenario. We consider a set of trusted scanners $S = (s_1, ..., s_n)$ that are deployed in various locations to detect unique identifiers of people. These scanners record all detected identifiers in a predetermined time, which we call an epoch e_i. At the end of an epoch, the scanners send their detections to a central processing server p that stores these detections, to continue recording during the next epoch. Authenticated clients $C = (c_1, ..., c_m)$ can subsequently query this server for the size of a crowd in a certain location. We distinguish two types of query:

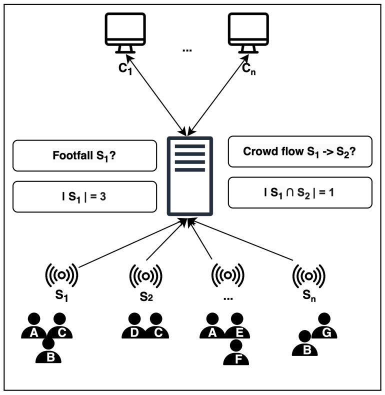


Fig. 1: Overview of the crowd monitoring system setup. Sensors $S = (s_1, ..., s_n)$ collect identifiers, process these and send them to a central processing server. Clients subsequently issue footfall or crowd flow queries to estimate the number of people at given locations.

- *Footfall*: the number of people in a single location during a single epoch.
- *Crowd flow*: the number of people traveling between different locations during a series of (not necessarily consecutive) epochs.

Formally, we define these queries as:

Definition 1. ***Footfall:*** *Let $D_{s,e}$ be the set of detections made by a scanner s during epoch e. The footfall provided by this scanner is the total count of unique detections $|D_{s,e}|$.*

Definition 2. ***Crowd flow:*** *Consider a set of scanners $S^* \subseteq S$, a series of (not necessarily consecutive) epochs $E^* = [e^*_1, e^*_2, \dots, e^*_k]$, along with a series of scanner-epoch pairs $SE = [(s^*_1, e^*_1), \dots, (s^*_k, e^*_k)]$, with $s^*_i \in S^*$ and $e^*_i \in E^*$. Let $D_{s^*_i, e^*_i}$ be the set of detections by s^*_i during epoch e^*_i. For a given set of scanner-epoch pairs SE, the size of the crowd flow is defined as the size of the intersection of these detection pairs: $|\bigcap_{(s^*_i, e^*_i) \in SE} D_{s^*_i, e^*_i}|$.*

Threat Model

The goal of this work is to provide both data protection and privacy preservation. In this model, we assume that all scanners are trusted, meaning that they can collect and process all identifiers. This is a realistic assumption as adversaries could circumvent the entire system by placing their own scanners to identify individuals. However, the other parties, i.e. the processing server and clients performing queries *cannot* learn the original identifiers (data protection), and should *not* be able to trace an individual with complete certainty (privacy preservation). We assume an honest but curious setting, where all parties follow the given protocol but will try to infer as much information as possible from the system. In this work, we focus mainly on the preservation of privacy, as we build upon other works that provide solutions only for data protection (see Section 4). Hence, for simplicity and better understanding, we describe our protocol in a plaintext variant and in Section 7, we describe how the protocol can be extended with homomorphic encryption to provide the additional data protection guarantees that other works have already introduced.

4 Background

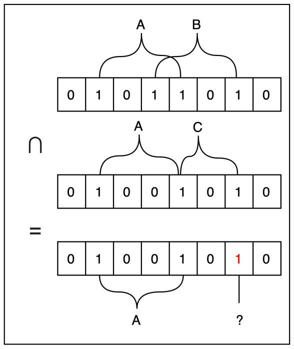


Fig. 2: Behavior of Bloom filter intersection. Intersecting two Bloom filters with ($m = 8$, $k = 2$) by bitwise multiplication, the result is not the same as the Bloom filter of their intersection.

In this paper, we propose methods to enhance privacy in crowd monitoring systems, based on the model introduced by [5], which guarantees data protection. Their system and threat models are similar to ours, except that we focus on privacy preservation on top of data protection. We aim to develop a crowd-monitoring system that preserves privacy and is capable of answering two key types of queries, footfall and crowd flow. Footfall is measured by the cardinality of a single detection set at a specific scanner and epoch, whereas crowd flow is determined by the cardinality of the intersection of detection sets collected from multiple scanners across different time epochs. Stanciu et al. [5] let each scanner store its detections during an epoch in a local Bloom filter. After the epoch, the Bloom filter is encrypted (homomorphically) and sent to a central server for further processing. The scanner subsequently discards all detections and creates a new, empty Bloom filter for the next epoch.

A Bloom filter [21] is a probabilistic data structure used to store whether elements are part of a set, consisting of an array of m bits, initially each set to 0, along with k different hash functions. When adding an element a, the k hash functions are computed on a. Each result points to one of the m array positions, which is then set to 1. To check whether an element is a member of a set, one needs to verify whether all the positions indicated by the k hash functions are set to 1. If we have two sets, A and B, represented as Bloom filters $BF(A)$ and $BF(B)$, respectively, the intersection $A \cap B$ is constructed by computing the bitwise AND operation on $BF(A)$ and $BF(B)$ (see Figure 2), where we assume that both Bloom filters have the same length m and use the same k hash functions. Due to the probabilistic nature of Bloom filters, the result approximates the intersection of A and B, yet is not guaranteed to be the same as $BF(A \cap B)$. Bloom filters may reveal false positive membership tests (false negatives cannot occur). This is because positions associated with an element can also be marked as 1 by the hashes of other elements. However, the parameters of the Bloom filter can be adjusted to achieve a desired probability of false positives (p) when the maximum number of elements in the set (n) is known.

We use Bloom filters to compute footfall and crowd-flow queries, considering the cardinality of the sets involved. The cardinality of a Bloom filter c can be estimated using the following formula, where t is the number of bits set to 1:

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

To perform a query, the system follows this procedure. Scanners encode their detections into Bloom filters and transmit them to the server at the end of each epoch. To ensure data protection, each Bloom filter is encrypted using a homomorphic encryption scheme before transmission. This allows the server to process the data—such as computing intersections for crowd flow queries—without learning the underlying identifiers. Upon receiving a query from a client, the server begins crafting a response by gathering the necessary data generated by scanners. For crowd flow queries, it generates a new Bloom filter through a bitwise AND of the corresponding Bloom filters. Then, it shuffles the positions to obscure any discernible patterns, transforming the Bloom filter into a randomized array of 0s and 1s (but without affecting the number of bits set to 1), before delivering the result to the client. The server is trusted to conduct the shuffling, as it is considered an integral part of the protocol, with the assumption that it will execute it accurately. After decrypting the shuffled Bloom filter, the client computes the desired statistical count using Eq. 1, as the count of 1s remains unaffected by the shuffling.

Stanciu et al. [21] proposed that the exclusive provision of statistical counts for crowd monitoring data could protect privacy, rather than using techniques such as anonymization, which removes personal identifiers. However, statistical outputs still present risks of individual identification. Using publicly available statistics, adversaries can conduct reconstruction attacks with the objective of identifying probable instances of individuals. These statistical reports can be linked to additional data using linkage techniques (see Vatsalan et al. [22]). Providing additional data about the monitored group becomes simpler when the group size is smaller, thus making the attack easier.

5 Methods

Regardless of the estimator we use for counting footfall or crowd flow, our goal is to have relatively high precision for high counts and low precision for low counts to protect privacy. Estimators operating on Bloom filters rely on the count of bits set to 1. Our key insight is to introduce controlled imprecision by giving a consumer access to a sample t^* of the actual number of bits t from which they can estimate the original count. The smaller the subset, the less precise the estimation, which helps obscure individual contributions to small counts. Specifically, we examine two methods to sample the number of bits that contribute to the estimate $t^* < t$: (1) sampling Bloom filter hash functions and (2) sampling detected identifiers. Bloom filters inherently introduce some inaccuracy due to false positives, and our methods leverage this property to further enhance privacy while maintaining utility for large-scale crowd monitoring.

5.1 Sampling hash functions

In this method, we use a standard Bloom filter of size m, using k hash functions. Each hash function h_i has a probability of q being used (and thus $1 - q$ of being skipped), deterministically determined by the element being inserted.[3] This determinism ensures consistency when sampling hash functions are applied to every identifier. This is crucial to maintain reliability when intersecting Bloom filters at different locations containing the same element. This method ensures that not all bits are used for inserting an item into the Bloom filter. By selectively including hash functions, we effectively reduce the number of set bits. However, given the reduced number of bits (which we call t^*), we cannot use Equation 1 as this formula assumes knowledge of the total number of bits that *would* be set without using our selection method. Therefore, we estimate the value of t from the subsample t^* using the following formula:

$$t = m\left(1 - \left(1 - \frac{t^*}{m}\right)^{\frac{1}{q}}\right) \tag{2}$$

When substituting t in Equation 1 with the Equation 2, we can simplify the computation for estimating the count of inserted items to:

$$c = -\frac{m}{k \times q} \ln\left(1 - \frac{t^*}{m}\right) \tag{3}$$

We have proved the derivations in Appendix 9.

5.2 Sampling identifiers

Consider a standard Bloom filter with a length of m and k hash functions. To reduce the number of bits to be used to estimate the size of the original set, each

[3] Using a seeded random function where the inserted element determines the seed.

identifier is inserted into the Bloom filter with probability q. It is crucial for this process to yield consistent decisions for each identifier to ensure the reliability of Bloom filter intersections when computing crowd flow. In other words, the decision to insert the detection of identifier ID is always the same, independent of when and where ID is detected. The total number of items inserted into the Bloom filter is estimated by modifying the original formula to account for the sampling identifier with probability q, as shown in Equation 3.

6 Evaluation

In this section, we evaluate how effectively our solutions estimate statistical counts for both footfall and crowd flow while enhancing privacy through controlled uncertainty, particularly in low-density crowd areas. While simple footfall counts can be directly obtained from Bloom filters without additional processing, crowd flows involve analyzing movement patterns across different locations or times, which requires processing intersected Bloom filters. We use generated detections [4] instead of real-world detections because they allow us to precisely control and measure privacy across a wide range of crowd flow scenarios, enabling us to comprehensively evaluate the precision of the system. Specifically, we compare the system's estimated statistical output with actual values obtained from controlled simulations. We performed $1,000$ repeated experiments using both the standard Bloom filter used in [5] and our sampling techniques, thoroughly evaluating their uncertainty levels across varying footfall and crowd flow sizes. To do so, we explore different combinations of parameters for Bloom filters and sampling methods. For the hash functions of the Bloom filters, we use MurmurHash3[5] for its efficiency and use of seed, providing an arbitrary number of hash functions.

The estimated number of identifiers in a Bloom filter may differ from the actual count for two reasons. First, the estimation is based on the probability of certain bits being set, which can vary as hash functions in Bloom filters have a probability of mapping different elements to the same position. Second, hash collisions during the intersection of multiple Bloom filters can cause bits to be marked incorrectly (see Figure 2), leading to discrepancies in the count for crowd flow queries. When we use sampling methods instead of inserting all crowd detections into Bloom filters, we expect significant differences between the actual count and the estimated count, especially with low actual counts and sampling does not provide a reliable representation.

We used the Root Mean Square Error (RMSE) to evaluate how closely the estimated count of inserted identifiers in the Bloom filter matches the actual count. It serves as a statistical measurement of the average deviation of predicted

[4] To have precise control over the size of detection sets and the count of shared identifiers between them, we generate sets of random unique identifiers to directly represent detections, mimicking pre-collected data.

[5] A. Appleby, *MurmurHash3*, 2016. Available at: `https://github.com/aappleby/smhasher/wiki/MurmurHash3`

values from actual values. In this research, RMSE quantifies the precision of the estimated count by measuring how closely it aligns with the actual count over $1,000$ repeated experiments.

$$RMSE = \sqrt{\frac{1}{x}\sum_{i=1}^{x}(c_{t_i} - c_i)^2} \tag{4}$$

Here, x represents the total number of experiments ($x = 1,000$), c_{t_i} represents the actual count of inserted identifiers, and c_i represents the estimated count. A lower RMSE value indicates better agreement between the estimated and actual counts, reflecting higher precision in the estimation process.

By employing RMSE, we can calculate the relative error using the formula:

$$\text{relative error} = \max\left(0, \frac{c_t \pm \text{RMSE}}{c_t}\right) \tag{5}$$

Here, c_t represents the actual count of inserted identifiers. Relative error represents the range within which the estimated value lies relative to the actual value, expressed as a ratio of the actual value. For a specific estimated value, if relative error $= 1$, then the estimated value is equal to the true value, which indicates that there is no deviation. If relative error > 1, then the estimated value is greater than the true value, suggesting an overestimation. If relative error < 1, then the estimated value is lower than the true value, indicating an underestimation. We select the maximum value between 0 and the computed error. Since we cannot predict a negative number of observations for footfall or crowd flow, estimating a negative value lacks a meaningful interpretation.

6.1 Footfall queries

To evaluate the impact of the proposed methods on the uncertainty of footfall query estimation, we use a standard Bloom filter configured for $1,000$ insertions ($n = 1,000$) and a false positive rate set at 0.01 ($p = 0.01$). Using the formulas $m = \frac{-n \ln p}{(\ln 2)^2}$ and $k = \log_2 p$ we determine the length of the Bloom filter ($m = 9586$) and the optimal number of hash functions ($k = 7$) based on the given parameters n and p. Subsequently, we ran experiments by inserting detection sets ranging in size from 1 to $1,000$ into the Bloom filter. Each experiment is repeated $1,000$ times with distinct sets of identifiers to ensure the robustness of our evaluation. Finally, we compute the Root Mean Square Error (RMSE) for footfall queries. In this section, we outline the configurations employed for each method and subsequently present the results obtained.

Sampling hash functions. We conducted experiments using different probability values, $q \in \{0.01, 0.02, 0.04, 0.0833, 0.1667, 0.3333, 1\}$.[6] In Fig. 3a, the observed trends and values align closely with our expectations, reducing the value

[6] Reflecting sampling rates of one out of $100, 50, 25, 12, 6, 3$, and 1, respectively.

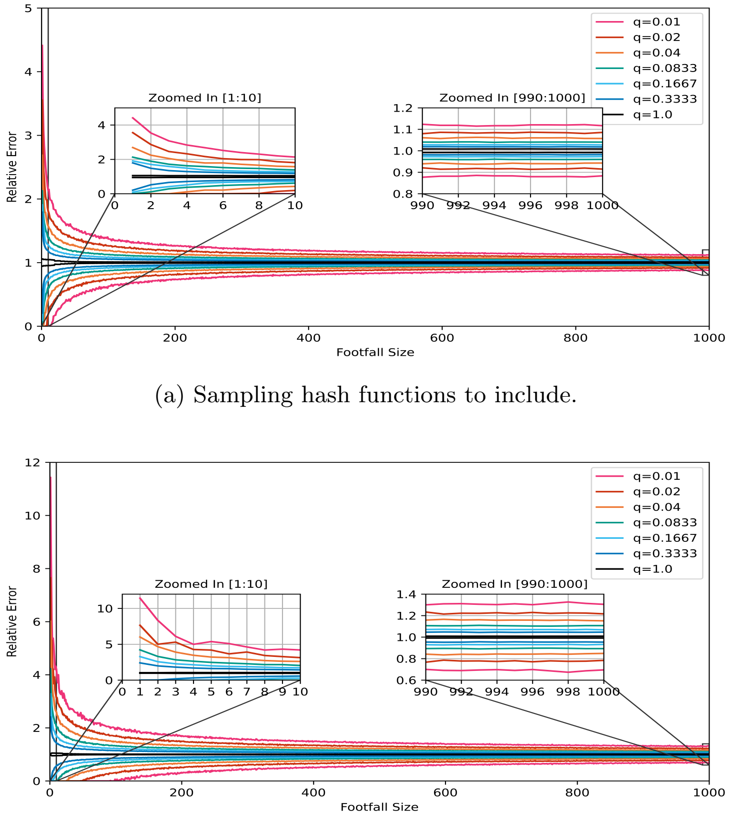
of q results in a higher relative error. This effect becomes especially notable when handling a small number of observations, while its impact diminishes for larger footfall sizes. For privacy-sensitive footfall sizes, particularly when the footfall size is 1, the system provides an intentionally uncertain output to enhance privacy. In this case, there is only a single detection, making direct estimation highly sensitive. When using a standard Bloom filter ($q = 1$), the relative error remains between 0.99 and 1.00. However, setting q to 0.3333 significantly increases the relative error, expanding its range to 0.20-1.79. When q is reduced to 0.01, the error range broadens drastically to 0–4.41, introducing uncertainty that effectively protects individual privacy. A relative error close to 1 suggests that the estimated count is highly precise. However, larger relative errors (e.g., 0–4.41 for $q = 0.01$ and footfall size 1) indicate greater uncertainty in the estimated count, making it difficult to determine the exact number of detections. This design choice prioritizes privacy for small footfall sizes. For larger footfall sizes, the relative error stabilizes. When the footfall size is $1,000$, a standard Bloom filter ($q = 1$) yields a relative error range of 0.99–1.00. With $q = 0.3333$, this range slightly shifts to 0.98–1.01, while reducing q to 0.01 results in an error range of 0.88–1.11. Since the relative error stays close to 1, the estimations remain reliable and useful for planning.

Sampling identifiers. In methods 5.1 and 5.2 described, the parameter q determines whether to include an item (regardless of whether we are dealing with a hash function or an identifier) in the Bloom filter. However, its effect on the Bloom filter may vary due to the context in which it is used. In method 5.1, q determines the probability of executing each hash function during the insertion of an item into the Bloom filter. This means that for each hash function, there is a chance of q that it will be applied. In method 5.2, q represents the probability of including an identifier in the Bloom filter. This probability directly influences the level of privacy and the number of identifiers stored in the Bloom filter. A lower q means that fewer identifiers are stored, improving privacy by reducing the likelihood of individual identification. In contrast, a higher q results in more identifiers being included, potentially compromising privacy by making individual representations more precise.

To assess the impact of detecting an identifier and to ensure a fair comparison with sampling hash functions, we use the same set of q values for our experiments. The results are shown in Fig. 3b. As expected, decreasing the value of q correlates with an increase in relative error. This increase becomes particularly noticeable when dealing with a limited number of detections, whereas its impact diminishes for larger footfall sizes. For example, when q is set to 1 (i.e., using a standard Bloom filter) with a footfall size of 1, the relative error ranges between 0.96 and 1.03. However, with q set to 0.3333, the relative error increases substantially, ranging from 0 to 2.40. Setting q to 0.01 further increases the error range between 0 and 11.40. The broader range of error observed with smaller footfall sizes offers advantages in terms of privacy concerns. In contrast, with a footfall size of $1,000$, the use of a standard Bloom filter produces a relative error range of 0.99 to 1.00.

When q is adjusted to 0.3333, this range changes to 0.95 and 1.04, while setting it to 0.01 causes the range to expand merely to 0.69 and 1.30.

When comparing the results of sampling hash functions (method 5.1) and sampling identifiers (method 5.2), we observe similar overall trends. In both approaches, reducing q significantly increases the relative error for smaller footfall sizes, providing greater privacy compared to the standard Bloom filter ($q = 1$) used in [5]. For $q = 0.01$ both methods introduce a substantial increase in relative error for small footfall sizes, reflecting their shared goal of obscuring individual detections. However, sampling identifiers (method 5.2) generally result in a wider error range compared to sampling hash functions (method 5.1), particularly when the footfall size is as low as 1. For larger footfall sizes, the difference between the two methods becomes less significant. Both converge to a relative error close to 1.



(a) Sampling hash functions to include.

(b) Sampling identifiers to include.

Fig. 3: Relative error of footfall queries. Footfall size ranging from 1 to 1,000. Bloom filter parameters: $n = 1,000$ and $p = 0.01$ using our proposed method.

6.2 Crowd-flow queries

Crowd flow is determined by the size of the intersection of multiple detection sets at different locations. In the system, these detection sets are represented by Bloom filters. To compute the size of the intersection of these Bloom filters, we use bitwise multiplication, which provides an estimate of the intersection of the underlying sets within the Bloom filters. It is important to note that the Bloom filter resulting from the bitwise multiplication operation on Bloom filters representing sets of detections is different from the Bloom filter representing the intersection of those sets of detections. Specifically, the same bits may be set in two Bloom filters, BF1 and BF2, due to different elements: one belonging only to BF1 and the other only to BF2. These bits will be incorrectly set to 1 in the bitwise multiplied Bloom filter (see Fig. 2). Consequently, the resulting number of set bits in the bitwise multiplied Bloom filter may not accurately represent the cardinality of the intersection of the two sets.

The number of bits set after performing a bitwise multiplication on BF1 and BF2, denoted as $t_\wedge$, is the sum of two terms:

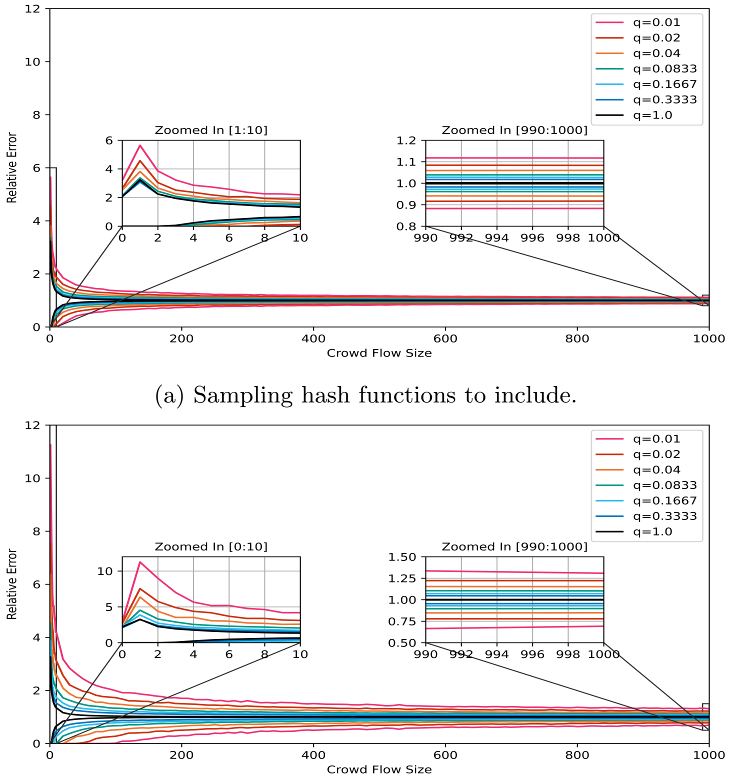
- The number of bits set due to the actual intersection of elements in both sets ($t_\cap$).
- The number of bits set caused by the hash collisions described above ($rbits$).

Consequently, the resulting Bloom filter from the bitwise multiplication of Bloom filters representing detection sets may differ from the Bloom filter representing their actual intersection. Thus, the formula used to estimate the number of common detections in sets tends to overestimate the actual counts. Papapetrou et al. [23] proposed an enhancement in estimating the number of common detections by incorporating not just the 1's in the resulting Bloom filter, but also those in the Bloom filters employed for bitwise multiplication. For t_1, t_2, and $t_\wedge$, representing the counts of bits set to 1 in two Bloom filters to be multiplied and in the resulting Bloom filter, respectively, the estimation formula is as follows:

$$c_\wedge = \frac{\ln(m - \frac{t_\wedge \times m - t_1 \times t_2}{m - t_1 - t_2 + t_\wedge}) - \ln(m)}{k \times \ln(1 - \frac{1}{m})} \tag{6}$$

To optimize two Bloom filters and assess the impact of our methods on crowd flow, we use the following setting. To reduce the impact of $rbits$ on the intersection and optimize the Bloom filters, we create a larger but sparser Bloom filter as suggested by Mitzenmacher [24]. Sparse Bloom filters have low entropy, which decreases the possibility of hash collisions. We design the Bloom filters for $10,000$ items and a false positive rate of 0.01. Then, we insert two randomly generated sets, each containing $1,000$ detections. We varied the number of common detections from 1% to 100% of the respective total detections of $1,000$ individuals.

To apply Eq. (6), a consumer would require knowledge of the responses to associated footfall queries alongside the crowd-flow query, allowing for the determination of parameters t_1 and t_2. This approach aligns with our system model, as consumers are allowed to initiate such queries and does not entail any additional computationally intensive operations. To prevent leaking information from other queries, one potential approach is to use fully homomorphic encryption (FHE) which would allow *the server* to perform calculations for crowd flow using parameters such as t_1, t_2, and $t_\wedge$ on encrypted data. The final output (crowd flow) remains encrypted, and only the consumer can decrypt it. This ensures that all intermediate data and computations remain secure, preventing accidental data exposure. We discuss this idea in more detail in Section 7.1. We employ equation 6 throughout the evaluation to estimate statistical counts on crowd flows, where we are especially interested in the error of our count compared to the actual size of the crowd flow. Adjustments to the formula are necessary for each method based on the sampling parameters.



(a) Sampling hash functions to include.

(b) Sampling identifiers to include.

Fig. 4: Relative error of crowd flow queries with a crowd size of 1,000 in both locations, and a crowd flow size ranging from 0 to 1,000. Bloom filter parameters: $n = 10,000$ and $p = 0.01$ using our proposed method.

Sampling hash functions. We conducted experiments with different probability values $q \in \{0.01, 0.02, 0.04, 0.0833, 0.1667, 0.3333, 1\}$. Analogous to Equation 3, we adjusted the crowd-flow estimation formula (6) to:

$$c_{\wedge} = \frac{\ln\left(\frac{(m-t_1^*)\times(m-t_2^*)}{m\times(m-t_1^*-t_2^*+t_{\wedge}^*)}\right)}{k \times q \times \ln\left(\frac{m-1}{m}\right)} \tag{7}$$

The results are shown in Fig. 4a. The results indicate that smaller values of q introduce greater uncertainty in crowd-flow estimation compared to using the standard Bloom filter with $q = 1$. Specifically, for small groups (e.g., crowd-flow size $= 1$), the relative error range is wider for smaller values of q. For example, when $q = 0.01$, the error can reach 5.64 for a crowd-flow size of 1, compared to an error of 3.23 when $q = 1$. By increasing the error in estimations, it becomes more difficult for an observer to determine the exact number of individuals in the crowd flow, thereby protecting individual privacy.

In larger crowds (e.g., crowd-flow size $= 1,000$), the relative errors decrease, but remain slightly higher for decreased values of q. For example, at $q = 1$, the narrow error range is between 0.998 and 1.002, while at $q = 0.01$, the relative error range is between 0.88 and 1.12. This shows that the system maintains accuracy and precision in estimating larger crowd sizes, ensuring that it is useful for planning and management.

Optimizing the value of q depends on the specific requirements of the application for privacy and utility. Smaller q values are beneficial for improving privacy in smaller groups, while maintaining higher q values ensures accurate estimation of crowd-flow sizes.

Sampling identifiers. For a fair comparison of the methods, we repeated the experiment using the same set of probability values q. Since we are sampling identifiers with a probability of q, we adjust the crowd-flow estimation formula (6) by dividing it by q. This adjustment makes the formula equivalent to (7).

The results (in Fig. 4b) show that we achieved our goal of introducing a higher relative error in small groups for privacy preservation by decreasing the sampling probability q. Specifically, for small groups (e.g., crowd-flow size $= 1$), the relative error is significantly higher for lower values of q, with an error of 11.26 for crowd flows of size 1 when q is 0.01 and 3.26 when q is 1. This higher error occurs because smaller groups have fewer identifiers, and sampling at lower probabilities reduces this even more. This makes the estimation less accurate and more random, which helps to protect individual privacy in less crowded areas. In contrast, for larger crowds (e.g., crowd-flow size $= 1,000$), the relative error remains relatively small across all values of q. Even at the lowest sampling probability of $q = 0.01$, the error is only 1.31, and decreases further as q increases, reaching 1.00 at $q = 1$. This indicates that the system remains reliable and precise for crowd flow size estimation in crowded areas, making it effective for planning and management purposes.

Overall, both sampling hashes and detected identifiers effectively balance privacy preservation in small groups with precise crowd flow size estimation in larger groups. In Section 7.4 we compare the two methods in terms of potential privacy breaches.

7 Discussion

7.1 Privacy enhancement using fully homomorphic encryption

Fully homomorphic encryption (FHE) [25] enables both addition and multiplication operations on encrypted data, allowing confidential computations required for privacy enhancement. By applying FHE, statistical computations, such as counting set bits in an Encrypted Bloom Filter (EBF), can occur without decryption. Unlike the current approach requiring server shuffling of EBFs, FHE eliminates the client's access to raw data. The result—a simple, encrypted count—is returned to the client, ensuring that the Bloom filters remain confidential, regardless of server behavior. While FHE provides significant privacy advantages, Bloom filters' size intensifies computational overhead. Segmenting Bloom filters into smaller blocks is a potential optimization for future research.

7.2 Identifiers for detection of individuals

Crowd-monitoring systems study pedestrian activity in public areas using sensors to gather identifiable data. Although our approach is independent of the chosen identifiers, some identifiers may be more suitable than others. In [5], the authors suggest using the MAC addresses of devices carried by individuals as identifiers. Recently, many Wi-Fi enabled devices have started using MAC address randomization. This means that devices replace their original MAC addresses with random ones during probe requests, according to the manufacturers' rules. The system can accurately estimate the number of devices, and thus the number of individuals, as long as each identifier belongs to a single device. However, some randomization techniques assign multiple identifiers to a single device, which can cause potential problems. For example, if a device uses several random MAC addresses within the same epoch, it can result in overestimating footfall. Additionally, if devices change their MAC addresses between epochs, it can easily cause underestimation of crowd flows, as the same device might be seen as different ones. To address these challenges, future work should focus on finding more reliable identifiers to accurately count footfall and crowd flow sizes.Although identifiers such as MAC addresses are generally subject to consent requirements, our system avoids this by ensuring privacy by design. Identifiers are processed only within trusted scanners, immediately encoded into encrypted Bloom filters, and then discarded. No raw or linkable data is stored or transmitted, and outputs are noisy and aggregated, preventing identification and need for consent.

7.3 Trade-Off between privacy and utility

We recommend optimizing hyperparameter q based on the specific application and sensitivity to privacy. This trade-off between privacy and utility can be adjusted to suit different scenarios. For scenarios prioritizing precise crowd monitoring with less emphasis on privacy, such as managing foot traffic during a large public event (e.g., a music festival), higher values of q provide sufficient data for accurate analysis of crowded areas. Conversely, in privacy-sensitive applications (e.g., monitoring pedestrian traffic in a quiet residential neighborhood), lower q values enhance privacy. This reduces the data granularity, ensuring residents' privacy while still providing approximate metrics for urban planning. By adjusting q, we can find the right balance between privacy and utility to suit the specific needs of the application.

7.4 Comparison of privacy-enhancing methods

We propose two methods to intentionally reduce the precision of estimates in sparse areas, each offering privacy and utility trade-offs. Method 5.2 applies the sampling probability q directly to the identifiers, meaning fewer identifiers are represented in the Bloom filter. While this approach increases privacy by reducing the contribution of individual identifiers, it also leads to higher errors as fewer bits are set in the filter, lowering its precision. Moreover, this method may not guarantee equal privacy for all identifiers. For instance, if certain identifiers are consistently included and the query result is zero, an adversary could infer that the individual associated with the included identifier is absent, thus compromising their privacy. On the other hand, Method 5.1 uses q to probabilistically include specific hash functions, effectively controlling the number of set bits in the Bloom filter. This approach achieves results comparable to Method 5.2 but avoids the issues associated with inconsistent identifier representation. By balancing precision and privacy, Method 5.1 ensures that errors remain manageable while minimizing the risk of privacy violations. Therefore, we recommend Method 5.1 as the preferred approach for improving privacy in crowd-monitoring systems.

8 Conclusion

We address the challenge of protecting privacy in crowd monitoring systems by limiting the amount of information that can be inferred about individuals. We achieve this by carefully limiting the granularity of data, ensuring that data about small groups is imprecise to prevent reidentification. In sparsely populated areas, this approach reduces privacy risks, while in densely populated regions, where accurate data is essential, our methods remain reliable and effective.

Our evaluation using simulated data confirms the effectiveness of our approach. By adjusting hyperparameters, we can balance privacy and utility, making the system adaptable to a wide range of real-world scenarios, from large public events to more privacy-sensitive environments. Future work should focus on refining these techniques, particularly exploring fully homomorphic encryption to enhance privacy while limiting computational overhead. In addition, the identification of more reliable and consistent identifiers will be crucial to improve the accuracy of crowd flow and footfall estimations. Our study contributes to the development of privacy-preserving crowd monitoring systems that protect individual privacy while delivering valuable insights for planning.

Acknowledgements

This research is funded by the Dutch Research Council (NWO) through the PERSPECTIEF Program P21-08 'XCARCITY'. We gratefully acknowledge their financial support.

9 Appendix

9.1 Footfall

Sampling hash functions. Starting equation:

$$t = m\left(1 - \left(1 - \frac{1}{m}\right)^{k}\right)$$

Substitute for t^*:

$$t^* = m\left(1 - \left(1 - \frac{1}{m}\right)^{kq}\right)$$

If we let:

$$\left(1 - \frac{1}{m}\right) = a, \quad \text{then:}$$

$$\frac{t}{m} = 1 - a^k \quad \Longrightarrow \quad a^k = 1 - \frac{t}{m} \quad \text{(Equation I)}$$

$$\frac{t^*}{m} = 1 - a^{kq} \quad \Longrightarrow \quad a^{kq} = 1 - \frac{t^*}{m} \quad \text{(Equation II)}$$

Apply Equation (I) into Equation (II), then:

$$1 - \frac{t^*}{m} = \left(1 - \frac{t}{m}\right)^{q}$$

Taking the q-th root of both sides:

$$\left(1 - \frac{t^*}{m}\right)^{\frac{1}{q}} = 1 - \frac{t}{m}$$

Rearranging:

$$\frac{t}{m} = 1 - \left(1 - \frac{t^*}{m}\right)^{\frac{1}{q}}$$

Finally:

$$t = m\left(1 - \left(1 - \frac{t^*}{m}\right)^{\frac{1}{q}}\right)$$

Starting equation:

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

Substitute:

$$t = m\left(1 - \left(1 - \frac{t^*}{m}\right)^{\frac{1}{q}}\right)$$

This leads to:

$$c = -\frac{m}{k}\ln\left(1 - \frac{1}{m}\left[m\left(1 - \left(1 - \frac{t^*}{m}\right)^{\frac{1}{q}}\right)\right]\right)$$

Simplifying:

$$c = -\frac{m}{k}\ln\left[\left(1 - \frac{t^*}{m}\right)^{\frac{1}{q}}\right]$$

Finally:

$$c = -\frac{m}{k}\cdot\frac{1}{q}\ln\left(1 - \frac{t^*}{m}\right)$$

Sampling identifiers. When sampling the identifier to insert them into the Bloom filter with a probability q, the standard BF estimator can be corrected by dividing estimated value by q:

$$c = \frac{c}{q}$$

9.2 crowd flow

Sampling hash functions. Let BF_1 and BF_2 be the Bloom filters of sets S_1 and S_2, respectively. The Bloom filters have length m and share the same k hash functions. $BF_\wedge$ is the Bloom filter created by a bitwise AND of BF_1 and BF_2. c denotes the cardinality of $S_1 \cap S_2$, and t_x denotes the count of the true bits set in the Bloom filter BF_x. In the paper by Papapetrou et al. [23], it is proved that:

$$t_\wedge = t_\cap + r_{\text{bits}} \tag{8}$$

$$t_\wedge = t_\cap + \frac{(t_1 - t_\cap) \times (t_2 - t_\cap)}{m - t_\cap} \tag{9}$$

In the standard Bloom filter, the expected number of bits set to 1 after inserting c elements using k hash functions in a Bloom filter of size m is given by:

$$t_\cap = m\left(1-\left(1-\frac{1}{m}\right)^{c\times k}\right)$$

Substituting this into the formula 9, we get:

$$t_\wedge = \frac{t_1 \times t_2 + m\left(1-\left(1-\frac{1}{m}\right)^{k\times c}\right)\times(m-t_1-t_2)}{m(1-\frac{1}{m})^{k\times c}} \tag{10}$$

Thus, in the standard Bloom filter, the expected number of common items in both sets (c) will be:

$$c = \frac{\ln\left(m - \frac{t_\wedge m - t_1 t_2}{m-t_1-t_2+t_\wedge}\right) - \ln(m)}{k\ln\left(1-\frac{1}{m}\right)}$$

When running each hash function with a probability of q, the expected number of bits set to 1 after inserting c elements is given by:

$$t_\cap = m\left(1-\left(1-\frac{1}{m}\right)^{k\times q\times c}\right)$$

Substituting this into the formula 9, we get:

$$t_\wedge = \frac{t_1 \times t_2 + m\left(1-\left(1-\frac{1}{m}\right)^{k\times q\times c}\right)\times(m-t_1-t_2)}{m(1-\frac{1}{m})^{k\times q\times c}} \tag{11}$$

Thus, the expected number of common items in both sets (c) is:

$$c = \frac{\ln\left(\frac{(m-t_1)\times(m-t_2)}{m(m-t_1-t_2+t_\wedge)}\right)}{kq\ln\left(\frac{m-1}{m}\right)}$$

Sampling identifiers. When sampling the identifier to insert them into the Bloom filter with a probability q, the standard BF estimator can be corrected by dividing estimated value by q:

$$c = \frac{c}{q}$$

References

1. Southworth, M.: Designing the walkable city. Journal of Urban Planning and Development 131(4), 246–257 (2005), https://doi.org/10.1061/(ASCE)0733-9488(2005)131:4(246)

2. Lai, Y., Kontokosta, C.E.: Quantifying place: Analyzing the drivers of pedestrian activity in dense urban environments. Landscape and Urban Planning 180, 166–178 (2018), `https://doi.org/10.1016/j.landurbplan.2018.08.018`
3. Martella, C., Li, J., Conrado, C., Vermeeren, A.: On current crowd management practices and the need for increased situation awareness, prediction, and intervention. Safety Science 91, 381–393 (2017), `https://doi.org/10.1016/j.ssci.2016.09.006`
4. de Montjoye, Y.A., Hidalgo, C.A., Verleysen, M., Blondel, V.D.: Unique in the crowd: The privacy bounds of human mobility. Scientific Reports 3(1), 1376 (2013), `https://doi.org/10.1038/srep01376`
5. Stanciu, V.D., Steen, M.v., Dobre, C., Peter, A.: Privacy-preserving crowd-monitoring using bloom filters and homomorphic encryption. In: Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking. p. 37–42. EdgeSys '21, Association for Computing Machinery, New York, NY, USA (2021), `https://doi.org/10.1145/3434770.3459735`
6. Musa, A.B.M., Eriksson, J.: Tracking unmodified smartphones using wi-fi monitors. In: Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems. p. 281–294. SenSys '12, Association for Computing Machinery, New York, NY, USA (2012), `https://doi.org/10.1145/2426656.2426685`
7. Schauer, L., Werner, M., Marcus, P.: 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. p. 171–177. MOBIQUITOUS '14, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL (2014), `https://doi.org/10.4108/icst.mobiquitous.2014.257870`
8. Bonné, B., Barzan, A., Quax, P., Lamotte, W.: Wifipi: Involuntary tracking of visitors at mass events. In: 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM). pp. 1–6 (2013), `https://doi.org/10.1109/WoWMoM.2013.6583443`
9. Cunche, M., Kaafar, M.A., Boreli, R.: Linking wireless devices using information contained in wi-fi probe requests. Pervasive and Mobile Computing 11, 56–69 (2014), `https://doi.org/10.1016/j.pmcj.2013.04.001`
10. Vanhoef, M., Matte, C., Cunche, M., Cardoso, L.S., Piessens, F.: 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. p. 413–424. ASIA CCS '16, Association for Computing Machinery, New York, NY, USA (2016), `https://doi.org/10.1145/2897845.2897883`
11. Stanciu, V.D., van Steen, M., Dobre, C., Peter, A.: k-anonymous crowd flow analytics. In: MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. p. 376–385. MobiQuitous '20, Association for Computing Machinery, New York, NY, USA (2021), `https://doi.org/10.1145/3448891.3448903`
12. Ferguson, N., Schneier, B., Kohno, T.: Cryptography Engineering: Design Principles and Practical Applications. Wiley (2010), `https://doi.org/10.1002/9781118722367`
13. Demir, L., Cunche, M., Lauradoux, C.: Analysing the privacy policies of wi-fi trackers. In: Proceedings of the 2014 Workshop on Physical Analytics. p. 39–44. WPA '14, Association for Computing Machinery, New York, NY, USA (2014), `https://doi.org/10.1145/2611264.2611266`

14. Marx, M., Zimmer, E., Mueller, T., Blochberger, M., Federrath, H.: Hashing of personally identifiable information is not sufficient. In: SICHERHEIT 2018, pp. 55–68. Gesellschaft für Informatik e.V., Bonn (2018), https://doi.org/10.18420/sicherheit2018_04
15. Kamp, M., Kopp, C., Mock, M., Boley, M., May, M.: Privacy-preserving mobility monitoring using sketches of stationary sensor readings. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 370–386. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), https://doi.org/10.1007/978-3-642-40994-3_24
16. Lim, R., Zimmerling, M., Thiele, L.: Passive, privacy-preserving real-time counting of unmodified smartphones via zigbee interference. In: 2015 International Conference on Distributed Computing in Sensor Systems. pp. 115–126 (2015), https://doi.org/10.1109/DCOSS.2015.13
17. Erlingsson, U., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. p. 1054–1067. CCS '14, Association for Computing Machinery, New York, NY, USA (2014), https://doi.org/10.1145/2660267.2660348
18. Ke, Y., Liang, Y., Sha, Z., Shi, Z., Song, Z.: Dpbloomfilter: Securing bloom filters with differential privacy. arXiv preprint arXiv:2502.00693 (2025)
19. Rusca, R., Carluccio, A., Casetti, C., Giaccone, P.: Privacy-preserving wifi-based crowd monitoring. Transactions on Emerging Telecommunications Technologies 35(3), e4956 (2024), https://doi.org/10.1002/ett.4956
20. Jiang, H., Li, J., Zhao, P., Zeng, F., Xiao, Z., Iyengar, A.: Location privacy-preserving mechanisms in location-based services: A comprehensive survey. ACM Comput. Surv. 54(1) (2021), https://doi.org/10.1145/3423165
21. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970), https://doi.org/10.1145/362686.362692
22. Vatsalan, D., Christen, P., Verykios, V.S.: A taxonomy of privacy-preserving record linkage techniques. Information Systems 38(6), 946–969 (2013), https://doi.org/10.1016/j.is.2012.11.005
23. Papapetrou, O., Siberski, W., Nejdl, W.: Cardinality estimation and dynamic length adaptation for bloom filters. Distributed and Parallel Databases 28(2), 119–156 (2010), https://doi.org/10.1007/s10619-010-7067-2
24. Mitzenmacher, M.: Compressed bloom filters. In: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing. p. 144–150. PODC '01, Association for Computing Machinery, New York, NY, USA (2001), https://doi.org/10.1145/383962.384004
25. Rivest, R.L., Adleman, L., Dertouzos, M.L., et al.: On data banks and privacy homomorphisms. Foundations of secure computation 4(11), 169–180 (1978), https://api.semanticscholar.org/CorpusID:6905087