Privacy-Preserving Crowd-Monitoring Using Bloom Filters and Homomorphic Encryption

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

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

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

Andreas Peter
a.peter@utwente.nl
University of Twente
Enschede, The Netherlands

Abstract

This paper introduces an architecture for crowd-monitoring which allows statistical counting for pedestrian dynamics while considering privacy-preservation for the individuals being sensed. Monitoring crowds of pedestrians has been an interesting area of study for many years. The recent prevalence of mobile devices paved the way for wide-scale deployments of infrastructures which perform automated sensing. Suddenly, people could be discreetly monitored by leveraging radio signals such as Wi-Fi probe requests periodically sent by their devices. However, this monitoring process implies dealing with sensitive data which is prone to privacy infringement by nature. While routinely performing their tasks, parties involved in this process can try to infer private information about individuals from the data they handle. Following privacy by design principles, we envision a construction which protects the short-term storage and processing of the collected privacy-sensitive sensor readings with strong cryptographic guarantees such that only the end-result (i.e. a statistical count) becomes available in the clear. We combine Bloom filters, to facilitate set membership testing for counting, with homomorphic encryption, to allow the oblivious performance of operations under encryption. We carry out an implementation of our solution using a resource-constrained device as a sensor and perform experiments which demonstrate its feasibility in practice.

CCS Concepts

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

Keywords

crowd-monitoring, pedestrian dynamics, statistical counting, privacy-preservation, Bloom filters, homomorphic encryption

ACM Reference Format:
Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. 2021. Privacy-Preserving Crowd-Monitoring Using Bloom Filters and Homomorphic Encryption. In *4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys'21), April 26, 2021, Online, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3434770.3459735

1 Introduction

Having a clear view of the dynamics developing within a crowd of pedestrians is highly important. When thinking about crowds of pedestrians, there are numerous situations that come to mind, ranging from commonly seen scenarios of buzzing pedestrian areas, congested transportation hubs, cramped touristic spots or popular shopping districts, to even massive gatherings such as large sports events or open-air festivals. Sought-after insights include information about how numerous the crowd is at certain points and what movement patterns are happening.

Measuring such dynamics is nowadays often done automatically, owing to the widespread adoption of carry-on devices, such as smartphones. These devices periodically emit so-called Wi-Fi probe requests, signals which are usually captured by a sensing infrastructure and later used for answering crowd-dynamics questions.

Such techniques have raised concerns regarding the privacy of individuals. This process of data capturing has shown to be so sensitive to privacy infringement that it is now clearly regulated, e.g., by the GDPR in the EU. According to the regulation, collecting such data without consent is strictly forbidden, unless it is used solely for statistical counting of people in a limited time and space. Even then, data must be anonymized or erased as soon as possible. As movement patterns can extend across large spaces and over long periods of time, it is not trivial to correlate information while discarding data in a timely manner. At the moment, organizations providing such services are facing challenges when doing their job of delivering insights to parties interested in understanding pedestrian dynamics. A solution is, thus, needed, which should take care of the privacy preservation aspects by construction while allowing these organizations to perform statistical counts.

In this paper, we introduce an architecture that offers statistical counting for pedestrian dynamics while protecting the data of individuals being sensed, at rest and during processing. Statistical counting is based on the composition of elementary queries, which take one of two forms: (1) how many devices are detected by sensor s during epoch e, or (2) how many devices have been detected by sensor s_1 during epoch e_1 and later by sensor s_2 during epoch e_2, where an epoch typically has a duration of several minutes. In our construction, the technical infrastructure is shielded from accessing data on pedestrians. At the same time, the statistical counting information about pedestrian dynamics is offered as a service to consumers interested in such scenarios, separating thus data gathering and processing from its usage. Producing such results actually boils down to performing counts without revealing what is being counted, i.e. computing the cardinality of either a set or an intersection of sets and nothing more than that.

EdgeSys'21, April 26, 2021, Online, United Kingdom

ACM ISBN 978-1-4503-8291-5/21/04.
https://doi.org/10.1145/3434770.3459735

As building blocks, we make use of Bloom filters (BFs), i.e. probabilistic data structures that allow for highly efficient set membership testing. Additionally, we employ homomorphic encryption (HE), which allows performing operations on the data under encryption. The combination of HE and BFs has been previously explored in different contexts as a promising solution for hiding the elements in a set while still being able to use them in computations. We investigate the use of these technologies in our crowd-dynamics setting, to perform set membership tests under encryption, as a means of measuring movement patterns across various locations and time intervals. Moreover, as sensors have to do cryptographic operations which are known to be costly, we implement a proof of concept and evaluate how it performs when using Raspberry Pi as a typical sensor device. Our results indicate that such a resource-constrained device can already operate well enough to be able to support the computation of multiple elementary queries within a reasonable time frame, which we anticipate is sufficient for many real-world composite queries ranging over several epochs.

2 System model

A system providing means for understanding pedestrian dynamics generally consists of a sensing infrastructure detecting people passing nearby, together with techniques to assemble meaningful information from the sensed data. Throughout the rest of the paper we refer to such a system as a *crowd-monitoring system* and we assume Wi-Fi as sensing technology. To model our solution, we start by describing the sensing infrastructure together with its expected behavior. Then, we model the situations relevant for pedestrian dynamics, as well as the entities involved in this process. Eventually, we formulate requirements for the system so that the privacy-sensitive data of individuals is protected.

2.1 Preliminaries

The core element of a crowd-monitoring system is the sensing infrastructure, i.e. a set $\mathcal{S} = \{s_1,...,s_n\}$ of sensors spread across a geographical area, capturing Wi-Fi signals from mobile devices carried by passers-by. For the clarity of exposition, we assume that sensors have nonoverlapping ranges, meaning that a device can be solely detected by a single sensor at a time.

When a sensor $s \in \mathcal{S}$ identifies a device in range, it reads its MAC address $a \in \mathcal{A}$ from the probe request, $\mathcal{A} \subset \{0,1\}^{48}$ representing the set of all existing MAC addresses. Each address reading belongs to an epoch $e \in \mathcal{E}$ corresponding to its timestamp t, such that $t_{start}(e) \leq t < t_{end}(e)$, where t_{start} and t_{end} mark the beginning and the end of an epoch and $\mathcal{E}$ denotes the set of all such epochs. A *detection* is, thus, uniquely identified by the 3-tuple (a,s,e), i.e. the device with MAC address a detected by sensor s during epoch e.

By $\mathcal{D}_{s,e}$ we denote the set comprising all the MAC addresses detected by a sensor s during an epoch e. By modeling $\mathcal{D}_{s,e}$ as a set, we assume that even if a device is detected multiple times within an epoch, we count it only once. Additionally, in our system we model the number of people in a specific area as the number of detected devices, this being the only information the system has. We are aware that the actual number of people may be different (e.g., because of people not carrying mobile devices) and we assume that a correction factor will be applied afterwards.

2.2 Counting for pedestrian dynamics

Relying on the detections previously introduced, various statistics can be derived. In particular, in our system we consider the following two situations:

- The crowd of people present in one place in a particular period of time, known as *footfall*
- The *crowd flow* of people traveling from one place to another

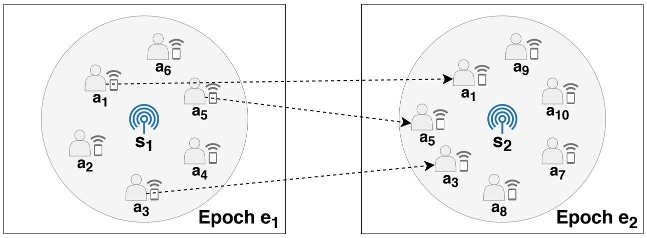

Figure 1: Situations encountered in pedestrian dynamics. Each individual square displays *footfall*, while the whole figure represents a *crowd flow*.

In Fig. 1, the two situations can be seen. On one hand, there is the crowd of people detected near each sensor during an epoch, representing *footfall*. On the other hand, there is a group of people from a crowd traveling from one place to another, a so-called *crowd flow*. Let us now formally define these situations in terms of counts.

Definition 1. Let $\mathcal{D}_{s,e}$ be the set of detections made by a sensor s during an epoch e within a crowd-monitoring system. We define the **footfall** in the area of sensor s during epoch e as the count obtained by computing $|\mathcal{D}_{s,e}|$.

Definition 2. For a collection of sets $\{\mathcal{D}_{s_1,e_1},...,\mathcal{D}_{s_n,e_n}\}$ representing detections made by sensors $s_1,...,s_n$ during epochs $e_1,...,e_n$ within a crowd-monitoring system, we define the **crowd flow** as the intersection $\bigcap_{i=1}^{n} \mathcal{D}_{s_i,e_i}$ over that collection. The size of the crowd flow following the corresponding path is the count obtained by computing $|\bigcap_{i=1}^{n} \mathcal{D}_{s_i,e_i}|$.

The crowd-monitoring system should function in such a way that it produces these two types of statistical counts. It should be also capable of responding to queries concerning them.

Having settled on what kind of information must be generated to support pedestrian dynamics, let us now model, from an architectural point of view, the entities taking part in the process. Please note that a depiction can be seen in Fig. 2.

Service Provider (SP). This is the entity that owns the sensing infrastructure. It is responsible for running the sensors and providing the pedestrian dynamics service. In order to offer this service, the SP should receive queries from interested parties and deliver the appropriate statistical counts as responses. In principle, the sensors could manage this process and collectively compute responses. However, the number of sensors can be high and the queries can be numerous, coming from multiple consumers and spanning across multiple locations and multiple epochs. Imagine that the sensors would have to store and exchange all this increasing amount of information, inherently leading to scalability issues. For this reason, in addition to the previously mentioned tasks, the SP should also run a separate service, not executed by the sensors, acting as a central manager. This service can be implemented, e.g., by a single or multiple cloud-based servers. For coherence, we are going to use the term *server* throughout the paper.

Consumers. Generally, these are public or private parties interested in understanding pedestrian dynamics. They are not part of the crowd-monitoring system, but they are the ones querying the system.

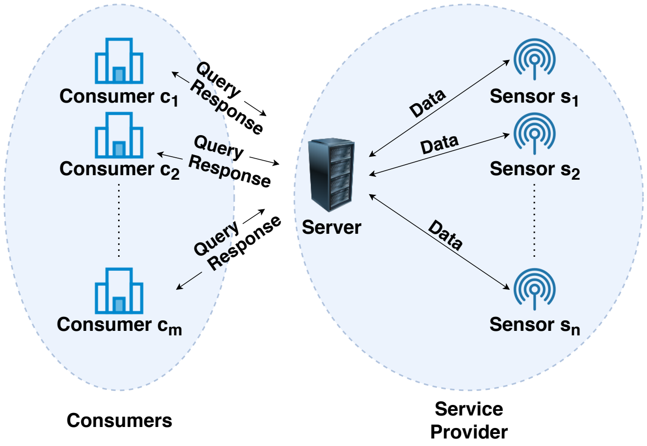

Figure 2: Service Architecture

2.3 Security requirements

Throughout the crowd-monitoring process, sensitive data is dealt with, such as unique identifiers of individuals, timestamps and locations where they are collected. It is highly important that a crowd-monitoring system is built in such a way that it does not leave any door open for tactics infringing the privacy of individuals. For this purpose, we are imposing a number of security requirements to be fulfilled while still being able to perform the statistical counts introduced in the previous subsection.

Sensors. We assume the sensors to be trusted. For this assumption to hold, we need to demand the sensors to be tamper-proof. On top of that, even if we trust the sensors, we should allow them to know only the least amount of information necessary for producing results. By that we mean that they should not see detections made by other sensors than themselves and they should not be aware of queries other than those in which they are involved. A final requirement for the sensors is that at the end of each epoch they should discard the detections made during that epoch, thus complying with common requirements of keeping data in clear as short as possible.

Oblivious server. The server should not be able to assemble any meaningful information from the arriving data; it should learn information needed only for running the protocol, such as sensors, consumers and queries. Hence, it should not be able to see information such as MAC addresses detected by sensors, results of the statistical counts or any other intermediary information related to individuals. Fulfilling this requirement keeps the individuals safe from honest-but-curious service providers, as well as protects their privacy in case of an attack on the service provider.

3 Our Construction

In this section we present our construction, following the system model introduced in Section 2. We start from the sensors, we present how they generate the required pedestrian dynamics information, and then we continue by adding several building blocks such that the security requirements are met.

The sensors are aware of the queries in which they are involved, as we have specified in the system model. This is a necessary condition so that they know how to behave during a specific epoch: they do only the work that they have been asked to do. In case of a footfall query, the sensors can simply gather detections in a set and, at the end of the epoch, they calculate the cardinality of that set, delivering the result to the server which forwards it to the intended consumer. The problem gets complicated when queries regarding crowd flows are asked, since answering them requires computing the cardinality of an intersection of sets coming from multiple sensors and epochs. We recall that we made the design decision of mandatorily discarding detections at the end of each epoch. As a consequence, we are now faced with the challenge of performing an intersection of sets without having the original sets any longer.

In our setting, computing the result of a crowd flow query as an intersection of sets is equivalent to performing a chained set membership testing across the sensors on the path of a crowd flow and carrying forward only the matching identifiers. The problem now transforms into finding a structure that allows us to perform membership testing without revealing what is stored in the sets themselves.

3.1 Bloom filters

A Bloom filter (BF) [5] is a space-efficient probabilistic data structure generally used for checking whether an element is a member of a set. It is represented as an array of m bits initially set to 0. Along with the BF, k different hash functions are also defined, each of them mapping a set element to one of the m array positions. A set element e is added in the BF by computing the k hash functions on e and setting the resulting positions to 1. Similarly, checking whether an element is a member of a set is done by verifying if the positions indicated by the k hash functions are all set to 1. For future reference, this is the same as multiplying the values found at those position and checking if the result is 1. BFs do not give false negatives, but false positives can be expected since the positions corresponding to an element could be also set to 1 by the hashes of other elements. However, research [7] has shown that the parameters of the BF can be easily tuned to obtain a desired false positive rate, which is acceptable for the domain of the application.

BFs are going to be used in our system to support the result computation for crowd flows. We will return to footfall queries later in this section. Instead of sets of detections, now the sensors involved in a crowd flow query make use of BFs in the following way. Leaving aside security requirements for a moment, the first sensor on a path generates a BF containing the detections made during an epoch and sends it to the server at the end of the epoch. Subsequent sensors download from the server the BF corresponding to the previous sensor on the path, check the BF for all the addresses they sense during the epoch concerning them, according to the query, and generate a new BF containing only the matched elements. The final sensor on the path computes the answer as the count of elements that were both sensed and found in the downloaded BF.

Up to this point we have a system delivering responses to both footfall and crowd flow queries. However, the solution is not complete as there are still a number of unfulfilled requirements:

- The advantage in terms of privacy provided by BFs relies on the fact that the hash functions are not made public. Considering that the MAC address space is easily enumerable [4], once the hash functions are known a BF can be brute-forced in limited time, thus revealing with high probability the MAC addresses it stores.
- When a sensor involved in a crowd flow receives a BF from the server, it can see which individuals are matched when performing set membership testing.
- The server sees the query results as we currently do not employ any mechanism to hide them. The same statement is true about the last sensors on the path of a crowd flow.

The answer to the problems mentioned above is combining BFs with an encryption scheme placing the server out of the game (it will store only encrypted information that it cannot decrypt) as well as blocking the sensors from gathering insights from the data they use when computing query responses. The only things to be seen in clear remain the results of the queries and solely by the specific consumers launching them. To satisfy our requirements, such an encryption scheme should bear the following properties:

- Encrypting the same value multiple times should deem different ciphers, so that no one can infer, by looking at the ciphers, which values are stored.
- It should allow multiplications under encryption, i.e. decrypting ciphers which result from multiplying other ciphers should be equal to the result of multiplying the unencrypted values.
- It should be asymmetric, such that encrypted data, both intermediary and responses, is meaningless for anyone accessing it but the intended consumer.

3.2 Homomorphic encryption

Homomorphic encryption (HE) [18] is a type of encryption which allows performing mathematical operations on encrypted data without a preliminary decryption. The results are encrypted too and they are the same as if the operations were performed on unencrypted data. There are multiple classes of HE schemes covering different types or frequency of operations. Partially homomorphic encryption (PHE) is a class of schemes which allows one type of mathematical operation, either addition or multiplication, to be performed on the encrypted data for an unlimited number of times. As we specified in 3.1, we are looking for an encryption scheme that allows multiplications under encryption, a property which holds for PHE.

ElGamal [13] is such a partially homomorphic cryptosystem allowing multiplications under encryption, which we choose to use in our construction. The algorithm is asymmetric, using a public key for encryption and a private key for decryption. Furthermore, the algorithm is probabilistic, involving randomness in the encryption process, so that encrypting the same value multiple times yields different and indistinguishable ciphertexts.

Let us now see how augmenting BFs with HE makes our construction complete. When consumers enroll in the crowd-monitoring system, they generate a public-private key pair and provide the public key to the SP, which distributes it to its sensors. Every time a sensor builds a BF as part of a query response for a consumer, before sending it to the server, it encrypts each position with the public key of that specific consumer[1]. Subsequent sensors take, for each device sensed, the encrypted values found at the positions indicated by the k hash functions, multiply each of them with an encrypted 1 (for rerandomization purposes) and copy the results at the exact same positions into a new BF. The rest of this new BF is filled with encrypted random numbers. Finally, the last sensor on the path performs set membership tests under encryption (Fig.3). It does so by multiplying, per device sensed, the k encrypted values together. The result of a multiplication could be either an encrypted 1, in case of a match, or an encrypted random number otherwise. Each result is then written in a collection, which will represent the response for the consumer. Please note that in a similar way the response for a footfall query is computed, by writing an encrypted 1 in a collection for each device detected during that single epoch. To find out a query response, a consumer iterates through such a collection, decrypts the ciphertexts and sums up the 1s.

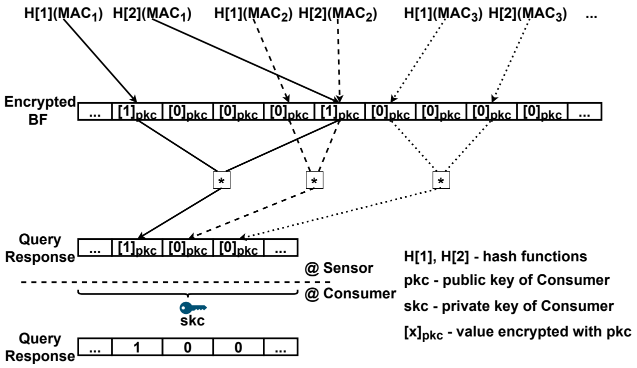
H[1](MAC$_1$)
H[2](MAC$_1$)
H[1](MAC$_2$)
H[2](MAC$_2$)
H[1](MAC$_3$)
H[2](MAC$_3$)
...
Encrypted BF
... [1]$_{pkc}$ [0]$_{pkc}$ [0]$_{pkc}$ [0]$_{pkc}$ [1]$_{pkc}$ [0]$_{pkc}$ [0]$_{pkc}$ [0]$_{pkc}$ [0]$_{pkc}$ ...
*
*
*
Query Response
... [1]$_{pkc}$ [0]$_{pkc}$ [0]$_{pkc}$ ...
@ Sensor
@ Consumer
skc
Query Response
... 1 0 0 ...
H[1], H[2] - hash functions
pkc - public key of Consumer
skc - private key of Consumer
[x]$_{pkc}$ - value encrypted with pkc

Figure 3: Statistical counting process under encryption, using an encrypted Bloom filter from another sensor and checking whether the detected MAC addresses are also members of that Bloom filter. The example considers k=2.

4 Experimental results

We have so far proposed a construction which fulfills, by design, the requirements for performing statistical counts on crowds while protecting privacy-sensitive data during processing and storage. The question now emerging is whether it is feasible to deploy this approach in practice and, if so, what kind of costs are to be expected.

In our solution, the core of data processing happens right at the edge, on the sensors, before reaching a server. Processing involves computing numerous hash functions, as well as encryption and multiplication under encryption operations which are known to be costly. Therefore, knowing that edge devices can have certain limitations in terms of capabilities, it came natural to aim for a proof of concept implementation directly on such a device, to get a clear picture of the problem.

We used throughout the experiments a Raspberry PI 4B, which uses a Broadcom BCM2711 SoC, with a 1.5GHz 64-bit quad-core ARM v8 Cortex-A72 processor, 8GB of DDR4 RAM memory, 16GB

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

microSD memory card and it is running Ubuntu 20.10 as OS. We did both serial and parallel implementations using C++11 threads for parallelization, MurmurHash3 [3] with different seeds for hashing and the SCAPI[2] library [12] for homomorphic encryption support. ElGamal is instantiated using the NIST P-256 elliptic curve [1].

BFs have the property that for each tested element, if all the positions indicated by the k hash functions are set to 1, then the element is probably present in the BF with a false positive rate of p. This is directly tied to the accuracy of the statistical counts as it is the only thing which can determine a count different than the one obtained by simply using the detections in clear. Recalling that BF parameters can be set such that a desired p is obtained, for a given maximum number of detections supported within an epoch n and a desired p we can compute the length m of the BF as $-n \ln p/(\ln 2)^2$ and the optimal number of hash functions k as $-\log_2 p$. We fix n to 1000, a sensor thus accommodating, per epoch, crowds carrying up to a thousand devices, and setup experiments according to Table 1. Alongside BF parameters we also display the resulting encrypted sizes.

Table 1: BF characteristics when n is fixed to 1000

p	m	k	size
0.0001	19173	13	13.01MB
0.001	14378	10	9.76MB
0.01	9593	7	6.51MB
0.1	4809	3	3.26MB

An edge device playing the role of an intermediary sensor in a query has to do the highest amount of work, which is up to $k*n$ hash computations and ElGamal multiplications, as well as m ElGamal encryptions. To this end, we show in Fig.4 the timing results for this specific scenario; similar setups for first and last sensors would lead to lower processing times. As a typical epoch length ranges from a couple of minutes to even hours, the results show that our solution can be safely deployed in practice using the considered hardware. For any trialed value of p, a sensor is able to perform computations for at least a couple of queries within the same epoch. At the same time, judging by the values in Table 1, the space complexity is not something that could raise concerns.

5 Discussion

Following common privacy by design principles, we deliberately require a consumer to announce the exact query he is interested in before the data collection starts. This leads to minimizing the data to be collected, using resources only when needed and avoiding unnecessary work. However, a consumer could realize that he is interested in the dynamics within a situation only after it happened. In principle, the system could be adapted to allow for such post-factum queries, but this will lead to further privacy challenges that we did not investigate.

Despite the fact that the most intensive computational part of our solution is bound to the sensors, we have shown that it is practically deployable even by using resource-constrained devices as sensors. By simply using more performant sensors which have, for example, more CPU cores, better performance can be expected as the data processing algorithm proved to be highly parallelizable due to the little dependency between operations. Still, there could be cases when a sensor must deal with an overly high number of simultaneous queries. For these specific cases and without lowering our privacy expectations, we plan to investigate the possibility of moving away operations under encryption to an external server with plenty of resources.

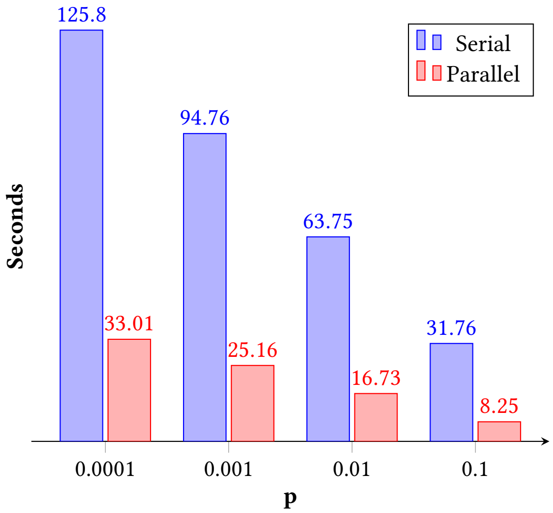

Figure 4: Most intense data processing scenario: an intermediary sensor detects the maximum number of devices (1000) during an epoch.

Depending on how large we consider the surface of a single location, more than one sensor might be needed to cover it. As a consequence, to count the people in that location we would have to compute a union of detection sets coming from sensors covering that surface, a *meta-sensor* as we denote. It is worth noting that computing such a union should happen no matter whether we are interested in counts concerning one or rather more locations. To accommodate such a setup, we plan to study different homomorphic schemes, as well as to consider additional assumptions regarding the communication within a meta-sensor. Along the same lines, new types of queries would be possible by supporting unions not only within the same location, but also across different locations.

Our approach is protecting only the collected privacy-sensitive sensor readings right after collection and during processing. The final end result, i.e. a statistical count, is then available in the clear to the designated consumer. In future work we will study whether these statistical counts allow for possible inference of privacy-invasive information (e.g., in edge cases where the monitored crowd consists of a single person) and, if so, what we can do about it (e.g., not allowing the system to produce any result in such cases).

6 Related Work

Performing Wi-Fi crowd-monitoring for pedestrian dynamics relies on Wi-Fi-enabled mobile devices which, by default, run in active scanning mode, periodically broadcasting messages known as *probe requests* in order to discover available networks [17]. When people carrying such devices walk around in a public space where a sensing infrastructure is installed, they can be automatically detected by the means of the messages being transmitted by their devices.

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

By analysing these detections, interested parties can learn valuable information about pedestrian dynamics, such as pedestrian crowd densities and flows [20], or different mobility patterns developing within the crowd [6].

All this is possible because the probe requests sent by smartphones contain, among other information, the MAC address of the device, thereby acting as a unique identifier. This poses a well-known problem in terms of privacy preservation since the address can expose individuals to unconsented tracking [8] or profiling [9]. As an effort to mitigate such identification, several hardware manufacturers implemented MAC address randomization, meaning that probe requests are sent with random instead of real MAC addresses. However, this has proven to be insufficient for protecting devices from reidentification, as shown in [21] and [15], getting back from a random MAC address to a unique identifier being possible through several techniques.

Organizations involved in Wi-Fi crowd-monitoring have also tried to address the problem, mostly by using pseudonymization. Hence, a pseudonym is calculated based on a sensed MAC address either by using a one-way hash function, a randomized allocation, or a deterministic encryption scheme. Due to the MAC address space being limited, such techniques are vulnerable to brute-force attacks, generating weak anonymized data [11]. Demir et al. [10] have even shown how such schemes used by most commercial solutions can be defeated using off-the-shelf equipment, an aspect reconfirmed as remaining a problem by the more recent work of Marx et al. [16].

Ultimately, research has been conducted by several scholars with the precise goal of protecting the privacy of individuals sensed by Wi-Fi systems. In [14], Kamp et al. introduce a solution based on linear counting sketches [19] for tracking the number of distinct persons that are present at a location, as well as reconstructing flows of persons between two or more locations. While the results look good when counting at a single location, the accuracy dramatically drops with the length of the flow when looking at pedestrian flows. Another interesting approach is based on differentially pan-private Bloom filters proposed by Allagan et al. [2]. Despite working well for large crowds, their proposal fails to deliver acceptable accuracies when dealing with smaller crowds. In contrast to these works, the solution proposed by us is not affected by the length of the flow, nor by the size of the crowds.

7 Conclusion

Measuring crowds of pedestrians can bring valuable insights for a wide range of domains. However, there are valid concerns regarding the privacy of individuals which are not sufficiently addressed by existing crowd-monitoring solutions. In this paper we introduced a crowd-monitoring system which can perform statistical counting for pedestrian dynamics and deliver results as a service to interested consumers while protecting the privacy-sensitive data of the individuals being sensed. We combined Bloom filters with homomorphic encryption in order to measure, in a privacy-preserving way, movement patterns across various locations and time intervals. As intensive operations are to be performed on sensors, we demonstrated the feasibility of such a construction by implementing a proof of concept while using a resource-constrained device as a sensor. Our experimental results showed that statistical counts can be computed within acceptable space and time bounds for any of the trialed BF probabilities of false positive.

References

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

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

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

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

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

[6] Bram Bonné, Arno Barzan, Peter Quax, and Wim Lamotte. 2013. WiFiPi: Involuntary tracking of visitors at mass events. In *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 1–6.

[7] Prosenjit Bose, Hua Guo, Evangelos Kranakis, Anil Maheshwari, Pat Morin, Jason Morrison, Michiel Smid, and Yihui Tang. 2008. On the false-positive rate of Bloom filters. *Inform. Process. Lett.* 108, 4 (2008), 210–213.

[8] Mathieu Cunche. 2014. I know your MAC Address: Targeted tracking of individual using Wi-Fi. *Journal of Computer Virology and Hacking Techniques* 10, 4 (2014), 219–227.

[9] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. 2014. Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing* 11 (2014), 56–69.

[10] Levent Demir, Mathieu Cunche, and Cédric Lauradoux. 2014. Analysing the privacy policies of Wi-Fi trackers. In *Proceedings of the 2014 workshop on physical analytics*. 39–44.

[11] Levent Demir, Amrit Kumar, Mathieu Cunche, and Cedric Lauradoux. 2017. The pitfalls of hashing for privacy. *IEEE Communications Surveys & Tutorials* 20, 1 (2017), 551–565.

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

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

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

[15] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. 2017. A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies* 2017, 4 (2017), 365–383.

[16] Matthias Marx, Ephraim Zimmer, Tobias Mueller, Maximilian Blochberger, and Hannes Federrath. 2018. Hashing of personally identifiable information is not sufficient. *SICHERHEIT 2018* (2018).

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

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

[19] Florin Rusu and Alin Dobra. 2007. Statistical analysis of sketch estimators. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 187–198.

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

[21] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. 2016. 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*. 413–424.