

# Real-time Crowd Counting based on Wearable Ephemeral IDs

Daniel Morales<sup>a</sup>, Isaac Agudo<sup>b</sup> and Javier Lopez<sup>c</sup>

Network, Information and Computer Security Lab, Department of Computer Science,  
ITIS Software, Universidad de Málaga, Málaga, Spain

**Keywords:** Pandemics, Crowd Counting, Bloom Filter, Privacy, SMPC, Ephemeral ID.

**Abstract:** Crowd Counting is a very interesting problem aiming at counting people typically based on density averages and/or aerial images. This is very useful to prevent crowd crushes, especially on urban environments with high crowd density, or to count people in public demonstrations. In addition, in the last years, it has become of paramount importance for pandemic management. For those reasons, giving users automatic mechanisms to anticipate high risk situations is essential. In this work, we analyze ID-based Crowd Counting, and propose a real-time Crowd Counting system based on the Ephemeral ID broadcast by contact tracing applications on wearable devices. We also performed some simulations that show the accuracy of our system in different situations.

## 1 INTRODUCTION


Crowd Density Estimation or Crowd Counting has always been a challenging task for historians. With the widespread growth of the urban scenario, new problems have emerged, with many implications on social, politics or economics areas. A clear example where Crowd Counting has had a lot of relevance is the estimation of participants in demonstrations. The estimated value in this type of events can involve changes on perception and alignment of people with respect to politic groups or social movements, so an accurate tool to achieve a real count can help to avoid manipulation and wrong perception (Janofsky, 1995). Other paradigm where Crowd Counting is present is urban management. When many people share urban resources, conflicts and accidents can emerge easily. If urban services like police, firemen or health entities have tools to anticipate to crowded events (rush hours, festivals...) they can handle unexpected events better and make more organized and efficient protocols for intervention. This paradigm has evolved to the next level on recent years, where the rise of the smart city paradigm has established new use cases for urban management services, where a massive number of devices is expected to be interconnected and interact with each other in an intelligent and automated way, processing data and statistics to make decisions and


providing services (Santana et al., 2020). However, another interesting, critical and novel scenario where Crowd Counting is an essential tool is pandemic management. Different pandemics have arisen along the human history and they have been faced in different ways. In this era, with the growth of the world's population and urban environments, having automated tools for both preventive and reactive decisions is fundamental. A clear example has been COVID-19, the disease generated by the virus SARS-CoV-2, which has been damaging the whole world since its origin in 2019. Many people have died, and keeping track of the virus spread has not been an easy task. For that reason, new tools that help avoiding the risk of crowd places and those tracking the spreading after positive tests have been developed in the last years, mostly based on  $CO_2$  sensors (Mumtaz et al., 2021) and smartphones (Troncoso et al., 2020).


In this work, we take advantage of Ephemeral Pseudo-Random IDs in an innovative way, using them to estimate crowd density in real time with an efficient and privacy friendly approach. Our solution can work both indoor and outdoors, in contrast with traditional approaches that are mainly based on indoor or short area environments, without any additional device apart from the citizens' smartphones.

The key contributions of this work can be summarized as follows:

1. We propose a generic architecture to perform Crowd Counting based on Ephemeral ID exchange which works on the application layer.

<sup>a</sup>  <https://orcid.org/0000-0002-6932-7437>

<sup>b</sup>  <https://orcid.org/0000-0002-2911-2300>

<sup>c</sup>  <https://orcid.org/0000-0001-8066-9991>

2. We define different scenarios where our solution fits, propose an implementation for a P2P environment, and formalize the main parameters that model the system behavior.
3. We simulate the proposed solution using synthetic mobility traces and analyze the impact of the parameters on the accuracy.
4. We analyze the security and privacy aspects of the proposed solution.

The rest of the paper is organized as follows. Section 2 summarizes some traditional techniques to achieve Crowd Counting. In Section 3 we briefly introduce the building blocks upon our proposal is built. Section 4 describes the general architecture to achieve Crowd Counting based on Ephemeral IDs and differentiates two application scenarios. Section 5 proposes a specific design for the P2P scenario, with implementation details and privacy protection based on a trusted proxy. In Section 6, we introduce a tiny prototype and describe some results obtained by simulation. Then, Section 7 proposes a modification to avoid a trusted proxy using SMPC. Finally, Section 8 summarizes some conclusions and future work.

## 2 RELATED WORK

Crowd Counting techniques have been traditionally approached from computer vision technologies, where images are processed to estimate the number of people in a scene.

A typical classification of techniques is made on three primary categories: (1) Detection-based approaches, (2) Regression-based approaches and (3) Convolutional Neural Networks (CNN)-based approaches. In the first category, the idea is to extract different features from images to detect complete or partial human bodies for counting (Felzenszwalb et al., 2010; Dollar et al., 2012).

In the second one, a direct mapping is established between extracted features and crowd estimation (or density map construction). It overcomes the difficulty for detection-based approaches to develop an accurate count on images where crowd is very close or there are considerable levels of occlusion and scene clutter (Chen et al., 2013; Idrees et al., 2013).

Finally, CNN-based approaches present the most accepted model because they achieve good accuracy when processing images with the previous mentioned problems (Walach and Wolf, 2016; Sindagi and Patel, 2017; Jiang et al., 2020; Wang et al., 2020).

In (Ryan et al., 2015), an evaluation is made over different models to perform Crowd Counting, as well

as a state of the art review and classification. Also, (Du et al., 2020) compares 14 algorithmic for Crowd Counting on images taken from drones.

Other recent works propose novel approaches, which reuse the existent models but apply them to more realistic and updated scenarios. That is the case of (Bailas et al., 2018), where authors test three different models in an edge computing setup. On the other hand, (Chen et al., 2019) propose an efficient CNN to estimate the Crowd Counting directly on embedded terminals.

Computer vision for Crowd Counting presents some problems. First, the need to set up specialized devices, with computing capabilities to handle the corresponding technique. Also, the devices need to be located strategically, because the area covered by a camera is limited. This problem is more significant in moving protests. Finally, there are privacy concerns too. Some techniques can obscure images and keep people privacy, but people have to trust the implementation of the system to work according to privacy and data protection regulations.

Other approaches are based on sensors. These are typically focused on indoor environments, where a specific number of sensors can be strategically positioned to keep track of people. A common approach is to use Passive Infrared (PIR) sensors (Wahl et al., 2012), but other types of sensors can be used. In (Kannan et al., 2012), the authors propose a model based on audio tones.

Some proposals go a step further and use sensors not to count people, but their devices. In the last years is not unusual to see most people carrying a smartphone almost all the time, above all they are in a working environment. In (Filippoupolitis et al., 2016), the authors propose a model based on a combination of distributed Bluetooth Low Energy (BLE) beacons and machine learning. A different but very common approach is to use WiFi technology. In (Zou et al., 2018), Channel Impulse Response (CIR) is used to discriminate individual paths in the time domain. With this approach, features can be extracted and then sent to a Crowd Counting classifier trained with machine learning techniques, like Naive Bayes or Support Vector Machines. Other approach is presented in (DePatla and Mostofi, 2018), where a Transmission-Reception model is used to estimate Crowd Counting based on the distortion that people do on Received Signal Strength Indicator (RSSI) values inside the building, or (Korany and Mostofi, 2021), also based on the WiFi channel state, but only for stationary crowds. Besides, there are also mobile network based approaches, like (Di Domenico et al., 2017), where authors propose crowd estimation based

on LTE signals.

Finally, the last type of techniques used to achieve Crowd Counting are based on distributed mobile applications. In (Danielis et al., 2017), authors propose a privacy preserving method where nodes can estimate the crowd size locally. This approach use an epidemically spreading model where devices can share with others the knowledge they have (shared and left nodes).

### 3 PRELIMINARIES

#### 3.1 Pseudo-Random Function

A pseudo-random function (PRF)  $F : X \times K \rightarrow Y$  is a deterministic algorithm that gets as input a key  $k$  and a piece of data  $x$  and outputs  $y = F(k, x)$ . Informally, we can define the security of a PRF as the warranty that, given a randomly chosen key  $k$ , the output  $y$  should look like a random function from  $X$  to  $Y$ , i.e., an adversary can only distinguish between the output from the PRF and the output from a random function with negligible probability<sup>1</sup>.

#### 3.2 Pseudo-random ID-based Contact Tracing

The work in (Chowdhury et al., 2020) presents an analysis about different applications that have been proposed for contact tracing. The main problem is to find a trade-off between usability and privacy. For that reason, different approaches have been proposed, based on a very wide set of technologies and privacy mechanisms. The different technologies covered by (Chowdhury et al., 2020) are the following: GPS, Bluetooth, WiFi, Cellular Network, RFID and NFC. Also, their system architecture is variate, and the computation can be carried out in a decentralized or centralized way. Privacy features heavily influence the architecture and performance of the system. Among those privacy techniques are Generic Multiparty Computation, Private Set Intersection Cardinality, Homomorphic Encryption or Blind Signature. However, on the most popular and widely used in many countries is the Ephemeral Pseudo-Random ID (Troncoso et al., 2020). The ID allows the system to compute calculations about users without a direct and personal identification of them. The advantage is that Pseudo-Random IDs are generally

<sup>1</sup>A negligible probability can be analyzed as a very small one, defined as  $p = 1/p(x)$ , where  $p(x)$  is a polynomial.

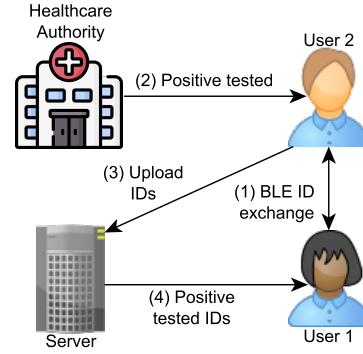


Figure 1: Ephemeral Pseudo-Random ID-based Contact Tracing.

computed using symmetric cryptographic techniques, so computations are lightweight, opposite to other mentioned techniques that require heavy asymmetric cryptographic computations. More specifically, the solution proposed in (Troncoso et al., 2020) generates a daily seed (Equation 1), where  $H$  is a cryptographic hash function.

$$SK_t = H(SK_{t-1}) \quad (1)$$

Each day, a rotating set of Ephemeral IDs are generated from  $SK_t$ , and those IDs are the ones shared among the closest peers. This is shown in Equation 2, where  $n$  is the daily number of IDs, obtained as  $n = (24 * 60) / L$ , being  $L$  a configurable parameter which sets the length of an epoch in minutes. Also,  $K_b$  is a fixed public string. The rotation mechanism is implemented to avoid the continue exposure of a user.

$$EphID_1 || \dots || EphID_n = PRG(PR(F(SK_t, K_b))) \quad (2)$$

Pseudo-Random IDs schemes (view Fig. 1), typically use the identifiers in a propagation phase, where close users exchange them using BLE advertisements. When a user is reported positive, her identifiers are uploaded to a central authority, where other users can download them as a list of positive tested identifiers. If any of the identifiers that a user has stored in her device correspond to any other downloaded from the positive list, the user is alerted with a exposure notification. More specifically, from (Troncoso et al., 2020), the positive tested user uploads the pair  $(SK_t, t)$ , which is distributed to other users who can compute the list  $\{EphID_i\}_{i \in [n]}$  and compare them with their stored EphIDs.

#### 3.3 Bloom Filters

A Bloom Filter (BF) is a privacy-preserving and space-efficient probabilistic data structure. It encapsulates a set of items into a bit array. To achieve this,

the bit array of size  $m$  is initialized to 0's and each item  $x$  is hashed to an integer in the range  $[1, m]$  with a set of  $k$  hash functions  $H = \{h_1, h_2, \dots, h_k\}$ . Each result  $h_i(x)$  points to a position of the bit array, which is set to 1. Given only the bit array is not possible to extract the previous items that have been inserted, but given an element one can check if it has been inserted (with high probability) comparing the 1's after computing the hashes. The false positive probability in Equation (3) is a key parameter in the design phase, because it relates  $k$  and  $m$  with the actual number of items in the filter  $n$ .

$$P_e \approx (1 - e^{-kn/m})^k \quad (3)$$

An interesting characteristic of BFs is that they can be aggregated with just the bit-wise OR operation. Also, given a single BF, an approximation of the set size can be conducted using Equation (4), from (Ashok and Mukkamala, 2014), where  $z$  represents the number of 0's in the BF.

$$|S| = \frac{\ln(z/m)}{k \ln(1 - 1/m)} \quad (4)$$

### 3.4 Adversaries and Security Definitions

To model and analyze the security warranties of the system, we must first define the capabilities of the adversary and the different aspects of the system that can be protected.

The adversary is defined as the abstract entity that interact with the elements of the system and can corrupt them to produce some malicious or unusual behavior. Plenty of adversary models have been proposed in the literature, but two of them remain as the most typical.

**Semi-honest Adversary:** the corrupted elements follow the protocol specification, but try to infer more data than allowed.

**Malicious Adversary:** the corrupted elements may deviate from the protocol, changing data, sending additional messages, or not sending them at all.

On the other hand, with respect to the system protection, we differentiate between two main properties, as different tools may be needed to achieve each of them.

**Privacy:** it refers to the property that the system provides to hide sensitive data from the adversary. Note that sensitive data may be dependent of the application, but it is typically related with user's personal behaviors or patterns exposure. For the proposed system, two types of sensitive data are identified: the pseudo-random IDs, which may lead to a linkability

attack to track users, and the geolocation data, which can be used maliciously to track users' movement.

**Security:** this notion refers to the warranties that the system works according to the specified protocol, avoiding incorrect results or leading to other threats.

## 4 GENERIC ARCHITECTURE FOR CROWD COUNTING

In this paper, we discuss Crowd Counting solutions that are based on IDs to estimate the count. The architecture can be proposed from different perspectives, depending on how the IDs are generated or used. We first define the general functionalities that must be present for the system to work.

**Pseudo-random ID Generation and Distribution:**

a Pseudo-Random ID is generated from a *seed* value. It identifies a device during a period of time, but the real identity of the device cannot be inferred from it. A Pseudo-Random ID can be persistent, if once generated it will always be the same value, and ephemeral, if it changes from time to time, specified by the length of an epoch. The seed value is determined by the application or technology employed. Also, the way to distribute the IDs is specific for each technology, and may be viewed as a lower layer of functionality which serves to our application (it typically will be a broadcast functionality). A solution may be to use periodic BLE advertisements, where the device can embed an ID generated by the application. Another possibility is to use the Pseudo-Random MACs enabled by WiFi, which provides this functionality to hide the real static MAC address of devices. In this case, the seed for the PRF is the network profile.

**ID Reporting:** this functionality comprises the generation of a location context based on the specific IDs. It can be a partial aggregation of the IDs received in a specific device or a report containing a location area for each device. As it will be later discussed, the reporting functionality presents a crucial role in the trade-off between accuracy and efficiency.

**Facilitator:** it refers to an endpoint that receives the reports emitted by the different reporters and perform the crowd count using the IDs or the location context information. This functionality is typically associated with a server with high capabilities.

As it was previously stated, the architecture can vary, depending on how the functionalities are deployed. For fixed location scenarios, e.g. indoor Crowd Counting, static nodes can facilitate the process. We identify two main architectures, depicted in Figure 2, that perform inverse solutions. Specifically, in Figure 2a, the static node broadcasts persistent IDs



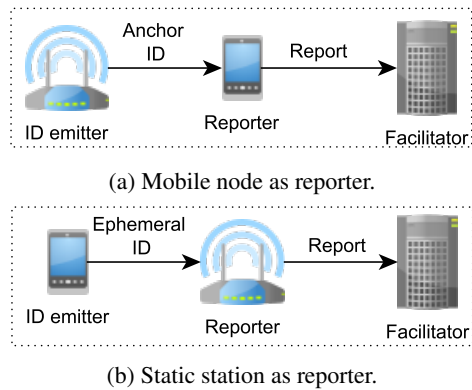


Figure 2: Generic architectures for Crowd Counting.

that serve as anchor IDs, e.g., a WiFi AP emitting the SSID. In this architecture, the mobile nodes send the anchor ID to the facilitator, which keeps track of the devices that are located inside the static node’s coverage area. On the other hand, in Figure 2b, the mobile nodes emit Ephemeral IDs and the static nodes are in charge of collecting them and reporting the facilitator. The second solution presents the advantage that the mobile node only has to broadcast the ID and nothing else, in contrast with the first solution where it has to read IDs from the environment and send them to the facilitator, increasing the resource consumption. It is also easier to embed a broadcast-only functionality directly on hardware chip-sets. In addition to that, using Ephemeral IDs reduces the tracking capabilities for an adversary along the system.

#### 4.1 Different Scenarios

The proposed functionalities are intended to be generic and cover the basic aspects needed for the system to work. However, they can be instantiated in different ways, depending on the specific scenario where the solution is to be applied. We identify and cover two of them, as it is shown in Figure 3.

**Enclosed Area:** Assume an scheduled event taking place in some fixed area, e.g., a free big concert, a demonstration, etc. For such a scenario, a straight solution may be to deploy one or some reporters that cover the whole place. Those reporters receive the IDs from the ID emitters, and send them to the facilitator. There is only one queried area for this scenario (the whole place), which is constantly recalculated by the facilitator. The ID emitters could be a small physical token, a wearable, a mobile phone, etc.

**Peer to Peer Scenario:** In this scenario, a very big area is covered, in such an extension that is not viable to set static reporters, e.g., a whole city. To achieve Crowd Counting, we propose to embed the

reporter functionality inside the same device that generates and emits the Ephemeral IDs. In this way, each single device will emit and receive IDs, and will send reports to the facilitator. This scenario has sense, e.g., in pandemics situations like the COVID-19 pandemic, where keeping track of contacts and avoiding crowds is a crucial matter to warranty security and health. In such a scenario, a contact tracing application can be reused as a tool for the Ephemeral IDs generation and management (Section 3.2).

## 5 IMPLEMENTATION DETAILS FOR THE P2P SCENARIO

We propose a specific design for the P2P scenario, where every ID emitter will also capture IDs and send reports to the facilitator. There are specific aspects to consider for this scenario that are not needed in a static one.

### 5.1 Spatial and Temporal Management

As a first aspect, the movement of the reporters introduce a constraint. For this scenario, location must be tagged for each ID, to allow location specific queries on the facilitator. If an individual report is sent for each received ID, the accuracy will be the best possible, as it will not introduce high errors on location data. However, this is not very realistic, due to the high number of connections that the reporters (low energy devices) may perform with the facilitator.

To fix that, we propose ID batching, i.e., receiving IDs during a specific interval of time and sending them together on a single report at once. This reduces the number of connections, but introduces a lower accuracy because of temporal and spatial errors.

Let’s assume that a user start receiving IDs when she is on location  $l_X$  and keep batching until she reaches  $l_Y$ , which can be considered “far” from  $l_X$ . If the report is tagged with  $l_Y$ , all the IDs will be aggregated on the facilitator for that location, modifying the real count. This presents the sampling interval as a critical parameter that must be taken into account for a trade-off between accuracy and efficiency. Algorithm 1 shows the pseudo-code for the ID batching procedure, where each batch is tagged with the mean value of  $(l_X, l_Y)$ .

The same approach could be used to handle time, but in this case it could be better to tag data with the facilitator reception time. The reason is that if the sampling interval is higher than 2 times the server historical time (more details in Section 5.4), then data are never used for queries.

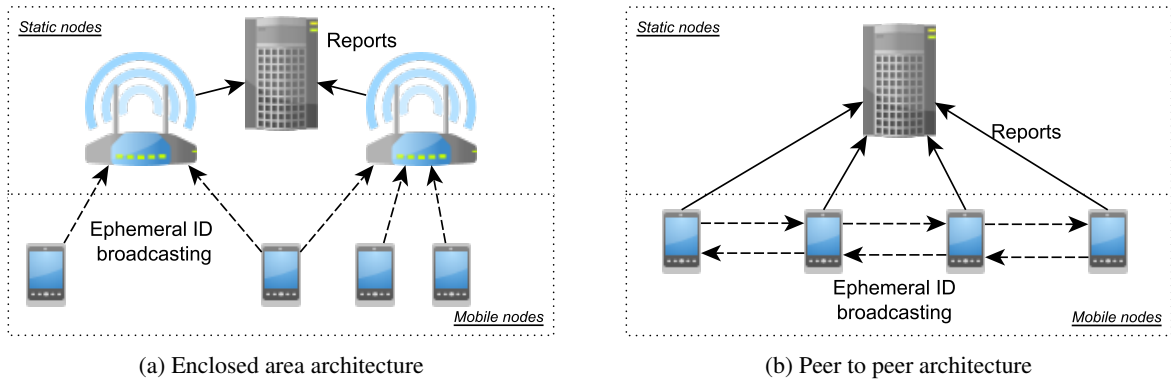


Figure 3: The communication architecture for the two proposed scenarios.

---

 Algorithm 1: Sampling Algorithm.
 

---

```

while true do
    Initialization : timer, loc, batch
    if read( $id_x$ ) then
        batch.insert( $id_x$ )
        timer.init(getTimeEnd(loc.getSpeed))
        locInit  $\leftarrow$  loc.getPosition
    end if
    while timer.notEnd do
        if read( $id_y$ ) then
            batch.insert( $id_y$ )
        end if
    end while
    locEnd  $\leftarrow$  loc.getPosition
    locMean  $\leftarrow$  mean(locInit, locEnd)
    batch.save(locMean)
end while
    
```

---

## 5.2 Raw ID vs BF ID Aggregation

We propose another modification to improve the management of IDs, both for batching and aggregation. Algorithm 2 shows a solution for the aggregation where the facilitator keeps a timer for each ID. This solution has the problem of the timer management, which can be huge for many IDs. Another solution could be computing the count for each query on demand, retrieving the saved batches which are not older enough. Following the latter solution, we propose an encapsulation of the IDs inside of a BF before sending the report to the facilitator, as shown in Algorithm 3. There are three main advantages for this solution: (1) it may reduce the size of each batch if the number of IDs is huge, (2) it reduces the count problem to bit-wise OR operations, and (3) it adds an extra layer of security, making the IDs opaque to the facilitator.

---

 Algorithm 2: Raw aggregation.
 

---

```

 $P_i \rightarrow$  Agg:  $\{ID\}^i$ 
    Agg:  $\forall id \in \{ID\}^i$ 
    if  $id \notin \{ID\}^{Agg}$  then
        save( $ID$ ), count = count + 1
    end if
    Agg:  $\forall id \in \{ID\}^{Agg}$ 
    if timer expired then
        delete( $ID$ ), count = count - 1
    end if
    
```

---



---

 Algorithm 3: BF aggregation.
 

---

```

 $P_i \rightarrow$  Agg: BF( $\{ID\}^i$ )
    if new query then
        Agg: count =  $\bigoplus_j BF_j$ 
    end if
    
```

---

## 5.3 Security and Privacy

We have to take privacy into account from the very beginning, not only because users are more and more aware about privacy, but also to respect privacy and data protection regulation. This system has to consider privacy from two points of view: related to user identification and related to user location.

User identification is made when an adversary relate a Pseudo-Random ID with a specific user, and can perform a tracing. This issue is solved with two layers of security. In the first place, if the Ephemeral IDs are correctly generated, i.e., they are pseudo-random and the seed is kept private, they cannot be linked to a specific user (contact tracing solutions also apply ID rotation to avoid continuous ID tracking). Secondly, applying BF encapsulation makes the IDs oblivious to the facilitator.

On the other hand, user location privacy is harder to achieve. There are many works that scope privacy on Location Based Services (Jiang et al., 2021), but it

does not exist consensus on the best solution.

The most typical solution tends to be  $K$ -Anonymity, usually implemented using cloaking techniques. This approach generalizes the area where the user is located until she blurs with at least  $K$  other users. The  $K$  users will report the same location to the server, so it is difficult for it to distinguish between them. The basic  $K$ -Anonymity schemes rely on a trusted server to perform the blurring (Francisco et al., 2003), but other solutions have been proposed to avoid a single trusted element, both in multi-server setting (Li et al., 2017), or in P2P environments (Chow et al., 2011). However,  $K$ -Anonymity is not good for our proposal, because privacy is directly related to location generalization. When high accuracy is required, the privacy provided by  $K$ -Anonymity will be low. This problem can be extended to other obfuscation schemes, like Differential Privacy.

Therefore, we propose a solution with a Trusted Proxy combined with Space Transformation (Figure 4), as a first solution to the problem, which will be extended later with some cryptographic aspects in Section 7 to avoid a single trusted element. Using a proxy has been proposed in the literature sometimes, overall for query privacy, e.g. (Singanamalla et al., 2021) propose an Oblivious DNS architecture. In our proposed solution, when a report reaches the Proxy, it maps the location to another space using a Pseudo-Random Function (PRF)  $f : L \rightarrow M$ , where  $L$  is the original location space and  $M$  is the mapping space. This function can be easily implemented using a symmetric cipher with location and the Proxy private key as inputs. Also, the mapping function can rotate periodically using different keys. So, the computation server receives the obfuscated and anonymous location reports and saves them into the database. The same procedure is applied to queries. To handle Space Transformation a space generalization must be made, but in contrast to  $K$ -Anonymity, it is not directly related to privacy, and small areas can be considered if the system allows for them. This solution still needs trust, like in Trusted Third Party  $K$ -Anonymity approaches, but the advantage is not just that the computation server can't track the position of users but it is even not able to determine which exact location is saving or being queried.

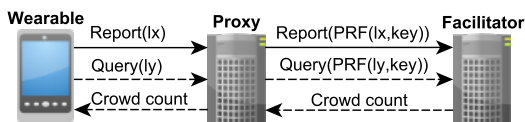


Figure 4: Trusted Proxy to achieve location privacy. The Key is only known by the Proxy.

The following sections will analyze security and privacy according to the party that is manipulated by the adversary.

### 5.3.1 Semi-honest Adversary

If every entity follows the protocol, there is only one leakage that can be produced, that is that the trusted proxy knows the geolocation data of the users. For that reason, this solution only achieves privacy on the trusted model.

### 5.3.2 Malicious Adversary

We analyze the malicious adversary from the different entities that can be corrupted.

**Malicious Reporter.** A malicious reporter can misbehave in different ways, e.g., sending dummy data to disrupt the estimated data on the facilitator side, both changing the location with a fake one or modifying the IDs which are sent inside the BF. The reporter can also stop sending notifications at all, deleting their presence on the server's counts, or on the contrary, perform denial of service (DoS) attacks sending notifications constantly.

**Malicious Proxy.** A malicious proxy can modify the received data from reporters, changing it with dummy data or not sending at all. It can also perform DoS attacks to the server, and send the location data without applying the PRF, allowing the server to read the location data and exposing users' privacy.

**Malicious Facilitator.** If both reporters and proxy are honest or semi-honest, the facilitator has no ability to retrieve the original location data or the pseudo-random IDs, as long as the PRF and BF remain secure. It can, however, change the counting data with totally random data, or cheat in the responses to make the application believe that a place is poorly crowded, when it is not true <sup>2</sup>.

## 5.4 System Characterization

In this section, we briefly describe the theoretical characteristics of the system. From now on, all assumptions are made within a specified location area, where people are not supposed to enter or leave it (a real scenario would imply more considerations). Equation (5) describes the probability for a device's ID to be in the system (for that location area), where  $\Delta_s$  is the previous time from a query reception where BFs are taken into account on the facilitator and  $T_r$  is

<sup>2</sup>Remark that, despite the cheating capacity, it cannot infer which real location belongs to each mapped location that has been received

the reporting period of a device. The first term is related to the device that reports its ID, but another term has to be added due to reports of other devices which include the scoped ID. This is represented with  $p_{other}$  and it relates the Bluetooth area range  $B_r$  and the area where devices are being considered  $A_u$ . The  $\min(\cdot, 1)$  function is used to limit the probability which increase theoretically with the influence of other BFs but is self-limited on a real implementation by the nature of adding BFs, where the same ID will not appear twice because of the bit-wise OR operation.

$$p_{dev} = \min\left(\frac{\Delta_s}{T_r} + p_{other}, 1\right) \quad (5)$$

Other consideration that must be made is that Ephemeral IDs change periodically. This affects the accuracy, but if the server tracks the IDs with low sampling rate, the added error is minimized. In Equation (6) the probability that an ID has changed is described, with  $T_{ID}$  being the period where the same ID is periodically sent from a device before changing it to the next one in the rotation.

$$p_{ID} = \frac{\Delta_s}{T_{ID}} \quad (6)$$

Finally, Equation (7) describes the estimated number of devices. It can be achieved combining the previous equations and relating it to the number of devices that are on the scoped area, determined by  $N$ . The variable  $\epsilon$  represents the error rate introduced by the cardinality estimation made on BFs, which is negligible.  $E_{inf}$  represents an increment of the estimated number of devices due to devices that are on surrounding areas close to those on the scoped area (the edge), with  $E_{inf} \geq 1$ .

$$|Dev| = (1 \pm \epsilon)(N \cdot p_{dev}(1 + p_{ID}))E_{inf} \quad (7)$$

## 6 SIMULATIONS

To give an approximate idea about the system performance, a tiny prototype has been implemented and some simulations have been performed. While it should not be taken as an exact example because it has some limitations (described below), it is still good enough to give the reader a general perspective of the expected results.

First of all, a general description is specified. The prototype is built upon two main different blocks: (1) the mobility scenario and (2) the facilitator implementation. Testing data related to moving users has never been an easy task, because it is difficult to access to

a wide amount of people simultaneously, and simulation is usually the choice, even with its limitations. In this work we have used a mobility trace generator (Mousavi et al., 2007), which generates mobility data based on some initial parameters, such as mobility model and simulation map. On the other hand, the facilitator server has been implemented using Python's Flask library<sup>3</sup>. The proxy has also been implemented, in regard to anonymize data received previously to send it to the facilitator. The latter saves data on a SQLite database<sup>4</sup>, and retrieves them on demand to handle queries. To test the real-time functionality a Python simulator has been developed which takes as input the mobility traces generated by (Mousavi et al., 2007) and outputs the tagged BFs in real-time which finally reach the proxy and the facilitator.

We will describe now the main limitations for the simulation. The first one is related to mobility models and it is that they are approximations to real traces, however accessing real traces on a controlled scenario is rarely possible. For this work we have tested the system with a Random Way Point model. The second limitation is related to data generation. The real interaction between users is very dynamic and it implies self timers for each one, in other words, arriving points for each user data is independent from others even if they have the same sampling interval. In this simulation data are statically generated and sent to the server at discrete time steps, so simulation data probably converge faster to more accurate results. Other important consideration is the way that BLE range has been modeled. In real contact tracing BLE range can be modified using the RSSI value as a threshold metric and keeping some counter to relate an ID received multiple times to a valid device nearby. The BLE considerations are generalized here just setting the range to 1 meter. As the simulation is discrete in time, an encounter is detected when two user's devices are closer than the BLE range and it is assumed that they exchange IDs in that time step and generate associated BFs.

There are two main parameters which determine system accuracy: temporal granularity, modeled with  $\Delta_s$  and spatial device density, modeled with  $d = \text{devices}/m^2$ . Fig. 5 shows a comparative on the same scenario with different values for  $\Delta_s$ , where *Actual Count* means the total number of devices that are on the queried area, *Actual Contacts* means the number of devices that are closer to others than the specified BLE range and *Estimated Count* means the estimation made on the server after adding BFs.

<sup>3</sup>Python's Flask library: <https://flask.palletsprojects.com/en/2.0.x/>

<sup>4</sup>SQLite: <https://www.sqlite.org/index.html>



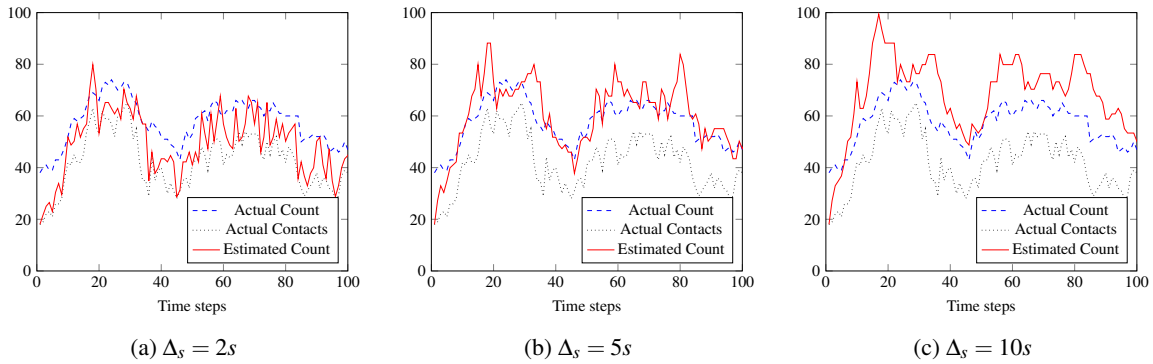


Figure 5: Simulation with 200 people in a 50x50 map with different  $\Delta_s$  values, speed 1 m/s and 20x20 area queried.

Table 1: RME values obtained from different simulations

Simulation	RME
Fig. 5a	0.1308
Fig. 5b	0.0620
Fig. 5c	0.1613
Fig. 6a	0.0422
Fig. 6b	0.3391

To measure the results we have used the Relative Mean Error of Equation (8) and obtained values are shown in Table 1.

$$RME = \frac{\sum_i^n |ActualCount_i - EstimatedCount_i|}{ActualCount_i} \quad (8)$$

In Fig. 5a Estimated Contacts are closer to Real Contacts, because they are the ones exchanging IDs, but alone devices are not taken into account. Fig. 5b shows the most accurate result, while Fig. 5c count more devices than there are, because the system takes into account devices that are not anymore in the scoped area. This comparative denotes the significance about selecting a precise value for  $\Delta_s$ . While a small value underestimates the number of devices, because older reports are not taken into account, a big value overestimates it, counting devices' IDs that are not in the queried location anymore.

On the other hand, density implications are shown in Fig. 6. The two of them are simulated with  $\Delta_s = 5$ , but while the first one gets a good accuracy when estimating the total number of devices within the scoped area, the second one doesn't. The reason is the poor number of ID exchanges carried on the second scenario, so it leads to conclude that the estimation is more accurate when more devices are closer between them, i.e., when people density grows.

## 7 IMPROVING THE TRUSTED PROXY WITH SMPC

Our first design presents a major caveat, which is the existence of a TTP. This solution is straightforward to avoid techniques that lower the precision of the measurements, but lacks of privacy. A distributed K-anonymity scheme will also involve higher communication rounds between the users, aspect we want to minimize because of the usage of low capability devices. For that reason, a solution to avoid loosing privacy and accuracy is to perform a distributed PRF. Distributed cryptography is a wide area of SMPC protocols (Zhao et al., 2019), where a group of nodes perform some cryptographic protocol on a privately-distributed piece of data, e.g., encryption or signatures. In our case, the PRF can be instantiated using deterministic AES (Harkins, 2008) or another PRF which has been implemented in a distributed way. The work in (Keller et al., 2017) proposes a solution for AES, where they achieve running-times of 0.928 ms for 2-party online phase and LAN setting.

For our proposed solution, the trusted proxy would be replaced by  $N \geq 2$  servers which will compute the distributed PRF on the inputs (secret shared values). The offline time does not matter because the precomputation is independent of inputs, and the online phase only introduces some small delays on the crowd size estimation, which are not so problematic (just a small temporal right shifting of the estimation made).

The main problem of this solution is that each user has to deliver one message to each server, increasing by  $N$  the communication and number of connections needed. To lower a bit that caveat, the infrastructure may work with a relay node, which cannot read the secret shares thanks to a Public Key Infrastructure (PKI), as shown in Figure 7.

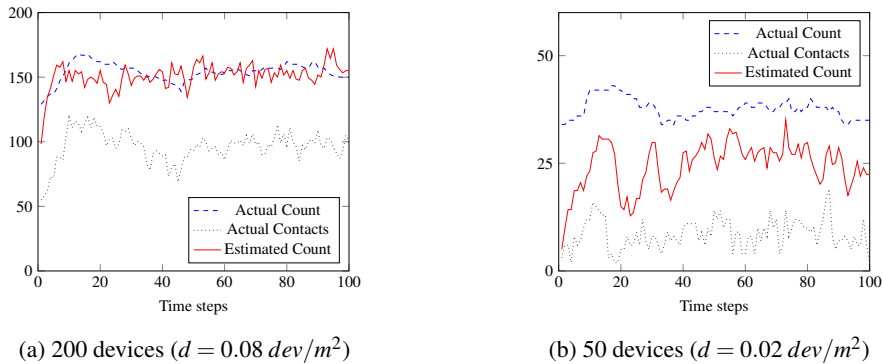


Figure 6: Simulation in a 50x50 map and 40x40 area queried, with different values for device density  $d$ .

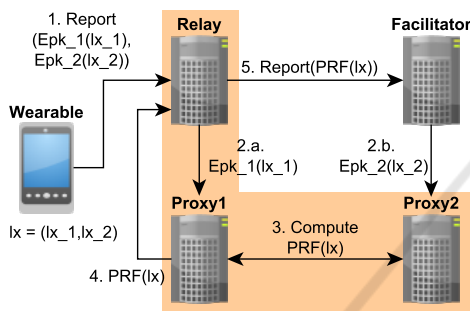


Figure 7: Architecture with SMPC PRF engine and relay node. The shaded area specifies the role of the “proxy”.  $E_{pk_X}(Y)$  means the encryption of  $Y$  using the public key of the proxy node  $X$ .

With respect to the adversary model, the setup now relies on the threshold scheme. As long as the adversary does not control more nodes than allowed for the SMPC scheme to remain secure, it cannot infer neither the PRF distributed secret key or the distributed input from the users. We can differentiate for the adversary as being the service provider or an external adversary. In the first case, it is trivial for the service provider to make  $n$  nodes to deviate, with  $n > t$ , however, it is clearly more difficult for an external adversary, which has not trivial access to the distributed computing cluster. In addition to that, privacy also remains even if the relay behaves maliciously, whenever the PRF and the PKI remain secure.

More difficult to achieve is general security, where every party can still introduce dummy data to the system. To solve this, more specific solutions have to be analyzed, like signatures or time-stamping.

## 8 CONCLUSIONS

In this work we have presented a novel real-time system to estimate Crowd Counting, based on Ephemeral

IDs and location data from wearable devices. This approach is built on the application layer, so it is easy to implement and it can be easily embedded, e.g., with a contact tracing application. Privacy preservation is a key aspect, but keeping a good trade-off between privacy and accuracy is not easy. While user identification is protected thanks to Pseudo-Random IDs and BFs, location data are more difficult. A Trusted Proxy has been used to provide anonymization and space transformation, and then modified to a distributed setup using Secure Multiparty Computation to avoid a single point of trust. Also, accuracy of estimation has been analyzed and it’s concluded that it depends on  $\Delta_s$  and  $d$  parameters. For a crowded area,  $\Delta_s$  is desirable to be relatively small and not overtake the actual devices count. If  $d$  is small, a small  $\Delta_s$  doesn’t work well, but it is not critical because the main objective is to detect crowds. However, it would be interesting as future work to test these conclusions on a scenario with real people, and also refine security aspects to prevent malicious behaviors.

## ACKNOWLEDGEMENTS

This work has been partially supported by the projects: SecureEDGE (PID2019-110565RB-I00) financed by the Spanish Ministry of Science and Innovation and SAVE (P18-TP-3724) financed by ‘Junta de Andalucía’. The first author has been funded by the Spanish Ministry of Education under the National F.P.U. Program (FPU19/01118).

## REFERENCES

Ashok, V. G. and Mukkamala, R. (2014). A scalable and efficient privacy preserving global itemset support approximation using bloom filters. In Atluri, V. and Per-

- nul, G., editors, *Data and Applications Security and Privacy XXVIII*, Lecture Notes in Computer Science, pages 382–389. Springer.
- Bailas, C., Marsden, M., Zhang, D., O'Connor, N. E., and Little, S. (2018). Performance of video processing at the edge for crowd-monitoring applications. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 482–487.
- Chen, J., Zhang, Q., Zheng, W., and Xie, X. (2019). Efficient and switchable CNN for crowd counting based on embedded terminal. 7:51533–51541. Conference Name: IEEE Access.
- Chen, K., Gong, S., Xiang, T., and Loy, C. C. (2013). Cumulative attribute space for age and crowd density estimation. page 8.
- Chow, C. Y., Mokbel, M. F., and Liu, X. (2011). Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments. *GeoInformatica*, 15:351–380.
- Chowdhury, M. J. M., Ferdous, M. S., Biswas, K., Chowdhury, N., and Muthukkumarasamy, V. (2020). COVID-19 contact tracing: Challenges and future directions. 8:225703–225729. Conference Name: IEEE Access.
- Danielis, P., Kouyoumdjieva, S. T., and Karlsson, G. (2017). UrbanCount: Mobile crowd counting in urban environments. In *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 640–648.
- Debatla, S. and Mostofi, Y. (2018). Crowd counting through walls using WiFi. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. ISSN: 2474-249X.
- Di Domenico, S., De Sanctis, M., Cianca, E., Colucci, P., and Bianchi, G. (2017). Lte-based passive device-free crowd density estimation. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6.
- Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. 34(4):743–761. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.
- Du, D., Wen, L., Zhu, P., Fan, H., Hu, Q., Ling, H., Shah, M., Pan, J., Al-Ali, A., Mohamed, A., Imene, B., Dong, B., Zhang, B., Nesma, B. H., Xu, C., Duan, C., Castiello, C., Mencar, C., Liang, D., Krüger, F., Vessio, G., Castellano, G., Wang, J., Gao, J., Abualsaud, K., Ding, L., Zhao, L., Cianciotta, M., Saqib, M., Almaadeed, N., Elharrouss, O., Lyu, P., Wang, Q., Liu, S., Qiu, S., Pan, S., Al-Maadeed, S., Khan, S. D., Khattab, T., Han, T., Golda, T., Xu, W., Bai, X., Xu, X., Li, X., Zhao, Y., Tian, Y., Lin, Y., Xu, Y., Yao, Y., Xu, Z., Zhao, Z., Luo, Z., Wei, Z., and Zhao, Z. (2020). Vidrone-cc2020: The vision meets drone crowd counting challenge results.
- Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. 32(9):1627–1645. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.
- Filippopolitis, A., Oliff, W., and Loukas, G. (2016). Bluetooth low energy based occupancy detection for emergency management. In *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on CyberSpace and Security (IUCC-CSS)*, pages 31–38.
- Francisco, S., Gruteser, M., and Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. page 31.
- Harkins, D. (2008). Rfc 5297 - synthetic initialization vector (siv) authenticated encryption using the advanced encryption standard (aes).
- Idrees, H., Saleemi, I., Seibert, C., and Shah, M. (2013). Multi-source multi-scale counting in extremely dense crowd images. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2547–2554. ISSN: 1063-6919.
- Janofsky, M. (1995). Federal parks chief calls 'million man' count low. *The New York times*, page 6–6.
- Jiang, H., Li, J., Zhao, P., Zeng, F., Xiao, Z., and Iyengar, A. (2021). Location privacy-preserving mechanisms in location-based services: A comprehensive survey. 54(1):4:1–4:36.
- Jiang, X., Zhang, L., Xu, M., Zhang, T., Lv, P., Zhou, B., Yang, X., and Pang, Y. (2020). Attention scaling for crowd counting. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4705–4714.
- Kannan, P. G., Venkatagiri, S. P., Chan, M. C., Ananda, A. L., and Peh, L.-S. (2012). Low cost crowd counting using audio tones. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, SenSys '12*, pages 155–168. Association for Computing Machinery.
- Keller, M., Orsini, E., Rotaru, D., Scholl, P., Soria-Vazquez, E., and Vivek, S. (2017). Faster secure multi-party computation of aes and des using lookup tables. In Gollmann, D., Miyaji, A., and Kikuchi, H., editors, *Applied Cryptography and Network Security*, pages 229–249, Cham. Springer International Publishing.
- Korany, B. and Mostofi, Y. (2021). Counting a stationary crowd using off-the-shelf wifi. *MobiSys 2021 - Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 202–214.
- Li, J., Yan, H., Liu, Z., Chen, X., Huang, X., and Wong, D. S. (2017). Location-sharing systems with enhanced privacy in mobile online social networks. *IEEE Systems Journal*, 11:439–448.
- Mousavi, S. M., Rabiee, H. R., Moshref, M., and Dabirmoghaddam, A. (2007). MobiSim: A framework for simulation of mobility models in mobile ad-hoc networks. In *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*, pages 82–82. ISSN: 2160-4894.
- Mumtaz, R., Zaidi, S. M. H., Shakir, M. Z., Shafi, U., Malik, M. M., Haque, A., Mumtaz, S., and Zaidi, S. A. R. (2021). Internet of things (iot) based indoor air quality sensing and predictive analytic—a covid-19 perspective. *Electronics*, 10(2).

- Ryan, D., Denman, S., Sridharan, S., and Fookes, C. (2015). An evaluation of crowd counting methods, features and regression models. 130:1–17.
- Santana, J. R., Sánchez, L., Sotres, P., Lanza, J., Llorente, T., and Muñoz, L. (2020). A privacy-aware crowd management system for smart cities and smart buildings. 8:135394–135405. Conference Name: IEEE Access.
- Sindagi, V. A. and Patel, V. M. (2017). CNN-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE.
- Singanamalla, S., Chunhapanya, S., Hoyland, J., Vavruša, M., Verma, T., Wu, P., Fayed, M., Heimerl, K., Sullivan, N., and Wood, C. (2021). Oblivious dns over https (odoh): A practical privacy enhancement to dns. *Proceedings on Privacy Enhancing Technologies*, 2021:575–592.
- Troncoso, C. et al. (2020). Decentralized privacy-preserving proximity tracing.
- Wahl, F., Milenkovic, M., and Amft, O. (2012). A distributed PIR-based approach for estimating people count in office environments. In *2012 IEEE 15th International Conference on Computational Science and Engineering*, pages 640–647.
- Walach, E. and Wolf, L. (2016). Learning to count with CNN boosting. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, volume 9906, pages 660–676. Springer International Publishing. Series Title: Lecture Notes in Computer Science.
- Wang, P., Gao, C., Wang, Y., Li, H., and Gao, Y. (2020). Mobilecount: An efficient encoder-decoder framework for real-time crowd counting. *Neurocomputing*, 407:292–299.
- Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C. Z., Li, H., and an Tan, Y. (2019). Secure multi-party computation: Theory, practice and applications. *Information Sciences*, 476:357–372.
- Zou, H., Zhou, Y., Yang, J., and Spanos, C. J. (2018). Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT. 174:309–322.