CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Location privacy without mutual trust: The spatial Bloom filter
Authors	Calderoni, Luca;Palmieri, Paolo;Maio, Dario
Publication date	2015-06-25
Original Citation	Calderoni, L., Palmieri, P. and Maio, D. (2015) 'Location privacy without mutual trust: The spatial Bloom filter', Computer Communications, 68(Supplement C), pp. 4-16. doi:10.1016/j.comcom.2015.06.011
Type of publication	Article (peer-reviewed)
Link to publisher's version	http://www.sciencedirect.com/science/article/pii/S0140366415002273 - 10.1016/j.comcom.2015.06.011
Rights	© 2015 Elsevier B.V. This manuscript version is made available under the CC-BY-NC-ND 4.0 license - http://creativecommons.org/licenses/by-nc-nd/4.0/
Download date	2024-11-13 04:51:45
Item downloaded from	https://hdl.handle.net/10468/4762

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Location privacy without mutual trust: the Spatial Bloom Filter

Luca Calderoni[a,*], Paolo Palmieri[b], Dario Maio[a]

[a] *Department of Computer Science and Engineering, University of Bologna, Cesena, FC, 47521 Italy*
[b] *Bournemouth University, Poole, Dorset, BH12 5BB, UK*

Abstract

Location-aware applications are one of the biggest innovations brought by the smartphone era, and are effectively changing our everyday lives. But we are only starting to grasp the privacy risks associated with constant tracking of our whereabouts. In order to continue using location-based services in the future without compromising our privacy and security, we need new, privacy-friendly applications and protocols. In this paper, we propose a new compact data structure based on Bloom filters, designed to store location information. The Spatial Bloom Filter (SBF), as we call it, is designed with privacy in mind, and we prove it by presenting two private positioning protocols based on the new primitive. The protocols keep the user's exact position private, but allow the provider of the service to learn when the user is close to specific points of interest, or inside predefined areas. At the same time, the points and areas of interest remain oblivious to the user. The two proposed protocols are aimed at different scenarios: a two-party setting, in which communication happens directly between the user and the service provider, and a three-party setting, in which the service provider outsources to a third party the communication with the user. A detailed evaluation of the efficiency and security of our solution shows that privacy can be achieved with minimal computational and communication overhead. The potential of spatial Bloom filters in terms of generality, security and compactness makes them ready for deployment, and may open the way for privacy preserving location-aware applications.

Keywords: Location Privacy, Bloom Filters, Secure Multi-party Computation

1. Introduction

Positioning systems are becoming more precise and more portable, and can now be easily embedded into smartphones and other personal devices. The combination of different positioning sources, such as signal strength for cellular phones, the visibility of wireless networks and more traditional satellite-based sources means that an address level precision can now be achieved even in low-cost and low-power devices. Satellite navigation and positioning systems are also seeing renewed interest, after years of stagnation: the deployment of a new global systems, Galileo, is currently being sponsored by the European Union, while regional systems such as the Chinese BeiDou (covering most of Asia), or the projected IRNSS in India are promising an even bigger increase in precision and capabilities.

The ability to know one's position with a certain degree of precision opened the way to the so-called location-aware applications, where users request personalized services based on their geographic position. Location-aware applications and services are now ubiquitous: from cell phone apps to intelligent car navigation systems, they are an integral part of our everyday life. In order to perform their task, location-aware applications usually require the user to disclose her exact position, in order to receive content and information relevant to the user's location. Examples of such location-aware services are local advertising, traffic or weather information, or suggestions about points of interest (PoI) in the user's surroundings [6]. Even existing services are now improved by the addition of location-based data: notable example are social networks [18] or retail distribution [11].

The ability to track a user's position raises however deep privacy concerns, due to the sensitive nature of location information. In fact, a number of potentially sensitive professional and personal information about an individual can be inferred knowing only her presence at specific places and times [1, 4]. Sensitive information such as religious beliefs, sexual preferences or

*Corresponding author. E-mail address: luca.calderoni@unibo.it

health conditions can be inferred by looking at the mobility trace of an individual, when he attends service at a church or a mosque, visits specific establishments or the practice of a specialized doctor. Even anonymized position data sets (not containing name, phone number or other obvious references to the person) do not prevent precise identification of the user: in fact, just four mobility traces may be enough to identify her. The more users disclose their data, the more providers are able to profile them in an accurate way. This is for instance the case discussed by Wicker in [45], where a marketing company database model is used in conjunction with anonymous mobile phone location traces. While we have become so used to smartphones and location-aware services that it would be very hard for a lot of us to give up on them, it is also reasonable to predict that in the coming years users will demand better privacy safeguards for their information with respect to the service provider [40], and more specifically for location information [24, 46]. The real challenge is therefore how to protect the user's privacy without losing the ability to deliver services based on her location [35].

A common application scenario of location-based services requires the service provider to learn when the user is close to some sensitive or interesting locations. This is the case, for instance, of "around-me" applications or security and military systems [6]. In this case, the location of the user should be kept private for as long as she is far from one of the areas of interest, and get disclosed to the service provider only when she enters one such area. A similar problem, known as *private proximity testing* has been studied in privacy research literature: Alice can test if she is close to Bob without either party revealing any other information about their location [30]. Narayanan et al. proposed a solution based on location tags (features of the physical environment) and relying on Facebook for the exchange of public keys [30]. His protocol was later improved in efficiency by Saldamli et al. [36]. Location tags and proximity tests are also used in [19], as a way of providing local authentication, while [47] presents a secure handshake for communication between the two actors in proximity. The security of the basic proximity testing protocol has been further improved in [31]. In [43], Tonicelli et al. propose a solution for proximity testing based on pre-distributed data, secure in the Universal Composability framework. Finally, the problem of checking the proximity in a specific time is addressed in [42].

In this paper we do not focus on proximity testing, but on a broader and more general problem: testing in a private manner whether a user is within one of a set of areas of arbitrary size and shape. By solving this problem and applying an intelligent conformation of areas, we can also solve the proximity testing problem (for one or multiple points simultaneously), and we are actually able to identify with some precision the distance of the user from the point of interest. Given the conceptual similarity of our problem with membership testing in sets, we base our solution on a novel modification of Bloom Filters (BF). Bloom filters are a compact data structure that allows to compute whether an element is a member of the set the filter has been built upon, without knowledge of the set itself [2]. Bloom filters have already been used in privacy-preservation protocols, and they are particularly suited to be used in conjunction with the homomorphic properties of certain public key encryption schemes [21].

1.1. Contribution

In this paper we propose a modification of Bloom filters aimed at managing location information, and we present two private positioning protocols for privacy-preserving location-aware applications. Although a preliminary version of the data structure was presented by Palmieri et al. in [34] (which forms the basis of the current work), in this paper we analyze the security and efficiency properties of the structure, and we provide a much greater insight on the usefulness of the construction to actual application scenarios, also by means of practical, real-world examples based on a test implementation.

The novel variant of Bloom filters we introduce, which we call *Spatial Bloom Filter* (SBF), is specifically designed to deal with location information. In particular, SBF combines multiple superimposed Bloom filters, in conjunction with an ad-hoc spatial representation, to provide a compact data structure for geographical information. Similarly to the classic Bloom filters, SBFs are also well suited to be used in privacy preserving applications, and we show this by presenting two protocols for private positioning. The protocols allow secure computation of location-aware information, while keeping the position of the user private: the only information disclosed to the provider is the user's vicinity to specific points of interest or his presence within predefined areas. At the same time, the areas of interest are not disclosed to the user. Therefore, in both settings we do not assume any trust between the parties. The first protocol is based on a two-party setting, where communication happens directly between the user of a location-based service and the service provider. A more complex scenario is defined in the second protocol, that involves

a three-party setting in which the service provider outsources to a third party the communication with the user. Both protocols achieve secure multi-party computation, where all parties have an interest in communicating, but want to keep their information private. In some cases, the privacy of the service provider can in fact be as important as that of the user: military and government applications are just the most immediate examples.

Following the definition of the spatial bloom filter and of the private-positioning protocols, we discuss the security and the computational cost of the proposed schemes, as well the probabilistic and storage properties of the SBF. In order to prove the readiness of the solution for actual deployment, we present the results of a prototype implementation of the filter creation and query routines. We base our tests on the geographic data of two real geographical regions: the metropolitan area of the city of Brussels, and Belgium. For both cases, we estimate optimal sizes for the filter and values for other important parameters.

Location privacy is a fundamental problem of the current age, where ubiquitous computing and unified communications are prevalent. The proposed solution is a solid step in the direction of more privacy-friendly services, and may enable privacy in both existing and future applications.

1.2. Related Works

With the recent introduction, and subsequent widespread diffusion of *location-based services* (LBS), the problem of preserving the privacy of the user with regard to his position arose. An early solution addressing this problem was presented in [16] by Gruteser and Grunwald, and consists in the application of k-anonymity to LBS: the location trace for each person should not be distinguished from at least $k-1$ other individuals, thanks to spatial and temporal cloaking of location and timing information. This is just one of the adopted metrics used to quantify privacy of a LBS. A comprehensive discussion of those metrics, including k-anonymity, is provided in [38], where the authors propose to preserve privacy by applying a distortion to the location information. Systems designed to protect location privacy are often referred to as *location-privacy protection mechanisms* (LPPM). Possible attacks to LPPM systems, and a proposal for a general framework able to evaluate the effectiveness of such systems is presented in [39].

While LBS's vary widely in terms of goals, a good number of them follow the model commonly known as *around-me* service. Here the user wants to find points of interests in his surroundings, based on his current position [6]. In order to achieve such goal, a location query is usually performed onto a remote server. In [20] the authors discuss privacy preservation with regard to the location queries used in this kind of service, and a taxonomy of location queries performed on the provider's server is also presented. A different privacy-preserving framework for location-based queries is proposed in [27]: the proposed solution relies on a trusted third party connecting the client with server. The problem of k nearest neighbor (k-NN) in location-based queries is addressed in a privacy-preserving manner in [26], where the authors propose using homomorphic encryption.

While around-me applications are usually designed for end-users, location privacy is also especially important in military and other government settings [12]. In [12] the protocol PRISM is presented. PRISM is designed to achieve privacy-friendly routing in MANETs, mainly for military purposes, using *group signatures*. Possible attacks against routing protocols, aiming at understanding the source location are discussed in [25]. Another growing field for LBS is represented by social networks. In [28] a flexible privacy-preserving location sharing system for mobile online social networks is discussed. Other sensitive applications requiring location data are alerting systems. In this context, Ghinita and Rughinis suggest that sensitive location information should be disclosed only when some conditions are met, such as when the user is within in some area of interests [15]. We follow the latter approach in this paper.

Unlike a majority of works discussing privacy in LBS, we do not focus on the protection of traces produced by the user's movements over time, and the related correlation attacks. Instead, this work aims at providing a cryptographic primitive that natively enables privacy in LBS, by preventing the creation of users' location-traces at all. In fact, the user exact position is generally concealed, and the service provider only learns the user's presence within a limited number of areas of interest. Moreover, contrary to the standard approach of focusing on the privacy of the user only, our solution is designed with both user's and provider's privacy in mind, as even areas of interest remain confidential.

1.3. Outline of the Paper

The paper is organized as follows: in Section 2 we provide useful notions and definitions that will be used later in the text, including the security model used in this paper. In Section 3 we introduce a spatial representation of Earth and we discuss how such a representation can be used to represent position as elements of a set. We

also present an algorithm to calculate distance from a point in the set-based setting. In Section 4 we define the Spatial Bloom Filter and we discuss important properties of the primitive, including an analysis of false positive probabilities. In Section 5 we propose two different protocols for the secure multi-party computation of position information in sensitive location-aware applications. Then, we discuss how the proposed schemes achieve private computation of location data without implying trust between the parties, and we analyze the security of the constructions. Two real-world examples are presented in Section 6, and are used as benchmark for a test implementation of the spatial bloom filters. Conclusions and ideas for future work are in Section 7.

2. Preliminaries

We introduce in the following some useful notions and definitions, that will be used later in the paper.

2.1. Bloom Filters

A Bloom Filter (BF) is a data structure that represents a set of elements in a space-efficient manner [2]. A BF generated for a specific set allows membership queries on the originating set without knowledge of the set itself. The BF always determines positively if an element is in the set, while elements outside the set are generally determined negatively, but with a probabilistic false positive error.

Definition 1. *We define a* Bloom filter $B(S)$ *representing a set* $S = \{a_1, \ldots, a_n\} \subseteq \{0,1\}^*$ *as the set*

$$B(S) = \bigcup_{a \in S, h \in H} h(a) \ , \tag{1}$$

where $H = \{h_1, \ldots, h_k\}$ *is a set of* k *hash functions such that each* $h_i \in H : \{0,1\}^* \to \{1, \ldots, m\}$*, that is, the hash functions take binary strings as input and output a number uniformly chosen in* $\{1, \ldots, m\}$*.*

A Bloom filter $B(S)$ can be represented as a binary vector b composed of m bits, where the i-th bit

$$b[i] = \begin{cases} 1 & \text{if } i \in B(S) \\ 0 & \text{if } i \notin B(S) \end{cases} \ . \tag{2}$$

The bloom filter is built as follows. Initially all bits are set to 0. Then, for each element $a \in S$ and for each $h \in H$ we calculate $h(a) = i$, and set the corresponding i-th bit of b to 1. Thus, m bits are needed in order to store b.

We test an element a_u against b to determine membership in S, that is, we verify whether $a_u \in S$ if

$$\forall h \in H, b[h(a_u)] = 1 \ . \tag{3}$$

If any bit in b that corresponds to a value output by one of the hash functions for a_u is 0, then $a_u \notin S$. If, instead, all the hashes map to bits of value 1, then $a_u \in S$ minus a false positive probability p determined by the number n of elements in S, the number k of hash functions in H and the maximum possible value m output by the hash functions (equal to the binary length of b) as follows:

$$p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \ . \tag{4}$$

This small false positive probability is due to the potential collision of hashes evaluated on different inputs, resulting into all bits associated to an element outside the originating set having value 1. As such, it is determined largely by k: if k is sufficiently small for given m and n, the resulting b is sufficiently sparse and collisions are infrequent. If we consider the approximation in (4), we can calculate the optimal number of hashes k as

$$\text{opt}(k) = \frac{m}{n} \ln 2 \ , \tag{5}$$

from which we can infer

$$m = \left\lceil -\frac{n \ln p}{(\ln 2)^2} \right\rceil \ . \tag{6}$$

However, the number of hashes also determines the number of bits read for membership queries, the number of bits written for adding elements to the filter, and the computational cost of calculating the hashes themselves. Therefore, in constrained settings, we may choose to use a less than optimal k, according to performance reasons, if the resulting p is considered sufficiently low for the specific application domain.

Bloom filter variants. Bloom filters have been extended to support advanced features over time.

Some remarkable variants of Bloom filters are *Counting Bloom Filters*, where the array of bits is replaced by an array of counters [13] and *d-Left Counting Bloom Filters* which basically pursue the same goal but save twice the space [5]. These variants allow to implement a delete operation on the filter without recreating it from scratch. *Compressed Bloom Filters* are designed to control precisely the memory consumption through a parameter z [29], while *Dynamic Bloom Filters* allow to create new filters in real-time as the originating set

varies [17]. However, none of these variants is suitable to store location data as multiple sets of areas nor to efficiently perform location queries upon them. The intuition to embed distance evaluation between the elements used to construct a Bloom filter was first presented in [23]: the paper uses locality-sensitive hash functions, and analyzes the performance under the Hamming metric. This approach, however, is not applicable to a geographic set, as the distance information is not naturally included in the value itself as it is in the case of elements that are numbers. The distance information needs therefore to be calculated with respect to a geographical representation, as we do in this paper.

Bloomier filters. Another variation of Bloom filters, the *Bloomier Filter* [8, 7], deserves a particular mention: a bloomier filter generalizes in fact a BF to store a binary function $f : S \rightarrow [0, 1]$ instead of a set. Bloomier filters thus allow to associate values with a subset of elements of the filter. At first glance, a bloomier filter could be mistakenly considered as very similar to the proposed SBF. In order to properly show their different features, we discuss relations of bloomier filters to the proposed spatial Bloom filters in Section 4.

Bloom filter applications. Bloom filters and some of their variants are used in various fields, including secure communications, network security, and secure multiparty computation [14].

Recently, a number of protocols and constructions for privacy-preservation based on Bloom filters have been proposed. Common application scenario for BFs are networking protocols. In particular, several protocols use them to perform message authentication [41] and node authentication [37] efficiently. Anonymous data transmission and anonymous route discovery was proposed in [9], while in [32] the authors introduced a system that uses BFs to prevent user movements detection based on to the tracking of RFID (Radio Frequency Identification) tags. BFs have also be used to cope with several kinds of malicious network behavior. A notable example of such use is a *Denial of Service* attack: filters are used to store information of each packet passing through a router, in order to enable tracebacking when required [22]. A comprehensive survey on Bloom filters variants, and their applications in network security was recently published by S. Geravand and M. Ahmadi [14].

2.2. Cryptographic Primitives

In part of our construction we use the homomorphic properties of encryption schemes. In general, a cipher has homomorphic properties when it is possible to perform certain computations on a ciphertext without decrypting it and, therefore, without knowledge of the decryption key. In particular, we say an encryption scheme is *additively homomorphic* when a specific operation $\boxplus$ applied on two ciphertexts $(\mathrm{Enc}(p_1), \mathrm{Enc}(p_2))$ decrypts to the sum of their corresponding plaintexts $(p_1 + p_2)$:

$$\mathrm{Dec}(\mathrm{Enc}(p_1) \boxplus \mathrm{Enc}(p_2)) = p_1 + p_2 \ . \tag{7}$$

There is additive homomorphism also when an operation on a ciphertext and a plaintext results in the sum of the two plaintexts. We have instead *multiplicative homomorphism* between an encrypted plaintext and a plaintext when an operation $\boxdot$ results into the multiplication of the two plaintexts:

$$\mathrm{Dec}(\mathrm{Enc}(p_1) \boxdot p_2) = p_1 \cdot p_2 \ . \tag{8}$$

An example of encryption scheme that is both additively and multiplicatively homomorphic is the Paillier cryptosystem [33]. In this case, the product of two ciphertexts will decrypt to the sum of their corresponding plaintexts (additive property), while an encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts (multiplicative property).

Private Hadamard Product. The Hadamard (or entry-wise) product of two vectors, one binary (owned by Alice) and one composed of natural numbers (owned by Bob), is performed in a privacy-preserving manner by Algorithm 1. The algorithm is private with respect to the input vectors, and only reveals the product vector to Alice. The security of the algorithm is based on the encryption of Alice's vector using a public key encryption scheme that is multiplicative homomorphic for operation $\boxdot$.

Algorithm 1 is analogous to the Secure Scalar Product algorithm presented in [21], and the same security considerations apply. A more conservative version of the algorithm requires Bob to multiply a randomly chosen prime number p, larger than any $\mathbf{y} \in \mathbf{Y}$, to each value in the vector, before performing the homomorphic multiplication. Alice can then obtain $\mathbf{X} \cdot \mathbf{Y}$ by calculating p using any greatest common divisor algorithm.

In general, we assume that the parties participating in the proposed construction do not deviate from the protocol, but gather all available information in order to try to learn private information of other parties. We are, therefore, in the semi-honest setting.

Algorithm 1: Private Hadamard product of an encrypted binary vector for a cleartext vector of natural numbers

Input Alice: $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{X} \in \{0,1\}^n$.
Input Bob: $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, $\mathbf{Y} \in \mathbb{N}^n$.
Output Alice: $\mathbf{X} \cdot \mathbf{Y}$.

1. Alice generates a public and private key pair using a multiplicative homomorphic encryption scheme, and sends the public key to Bob.
2. Alice sends to Bob the ciphertext vector $\mathbf{E} = (\mathrm{Enc}(\mathbf{x}_1), \ldots, \mathrm{Enc}(\mathbf{x}_n))$.
3. Bob computes the vector $\mathbf{C} = (\mathrm{Enc}(\mathbf{x}_1) \boxdot \mathbf{y}_1, \ldots, \mathrm{Enc}(\mathbf{x}_n) \boxdot \mathbf{y}_n)$ and sends the result to Alice.
4. Alice uses her secret key to decrypt $\mathbf{C}$ and obtains $\mathbf{D} = \mathrm{Dec}(\mathbf{C}) = \mathbf{X} \cdot \mathbf{Y}$.

2.3. Privacy Definitions

As our construction relies on Bloom filters, the privacy model is defined accordingly. Given the nature of Bloom filters (a data structure based on sets and designed for membership queries), we propose a privacy model that is, in a similar way, based on sets and aimed at preserving privacy in (location-based, that is, geographical) membership queries. Location-privacy is therefore evaluated according to two location-sets: the first set contains a limited amount of regions monitored by the provider while the second contains all of the remaining regions on the Earth's surface.

Privacy Model. Given two sets of geographic regions A and B, and a location-based protocol between a user and a service provider, the user's location remains concealed to the provider while the user is in a region contained in B. On the contrary, if the user is within a region contained in A, the provider learns, at least, that the user is in A, and, at most, the region in A in which the user is, but never the user's exact position.

In the following, we introduce the security model we adopt for the proposed construction.

Security Model. We assume the parties are *honest-but-curious* [10], that is, the parties will follow the protocol but try to learn additional information about other parties private data.

3. Spatial Representation

The construction we present in this paper is based on a novel variant of BFs aimed at managing location information. Since BFs are constructed over finite sets of elements, we need to represent location information – that is, a geographical position – as an element that is part of the finite and discrete set of all possible positions. Therefore, instead of considering a location as a point, we divide Earth's surface into a set of distinct regions, and we identify a position as the corresponding element in this set.

0.001 degrees	**lng**		**lat**	
equator	111.32 m	~	111.00 m	
23th parallel N/S	102.47 m	~	111.00 m	Cuba
45th parallel N/S	78.71 m	~	111.00 m	Italy
67th parallel N/S	43.50 m	~	111.00 m	Alaska

Table 1: Some reference values of accuracy using three decimal places for coordinate representation.

Considering that we can set the dimension of such regions to an arbitrarily small size, there is no loss in the precision of the location information. In particular, we do not use this approach in order to obfuscate or partially hide an exact position: on the contrary, we are interested in retaining a precision as high as the one allowed by the location sensor used in the specific application.

The most natural spatial representation for Earth is the standard geographic coordinate system. In the geographic coordinate system every location on Earth can be specified by using a set of values, called coordinates. Standard coordinates are *latitude*, *longitude* and *elevation*. For the purposes of this work we focus on longitude and latitude only, as the combination of these two components is enough to determine the position of any point on the planet (excluding elevation or depth). The whole Earth is divided with 180 parallels and 360 meridians; the plotted grid resulting on the surface is

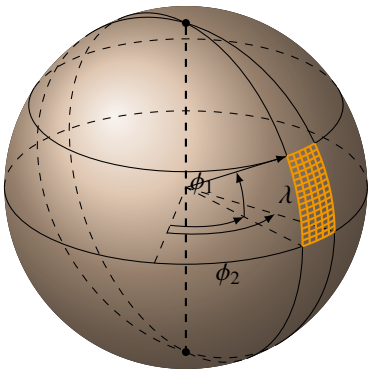


Figure 1: An example of the planet's surface and the grid plotted on it. ϕ_1 and ϕ_2 are longitude values while λ is a latitude value.

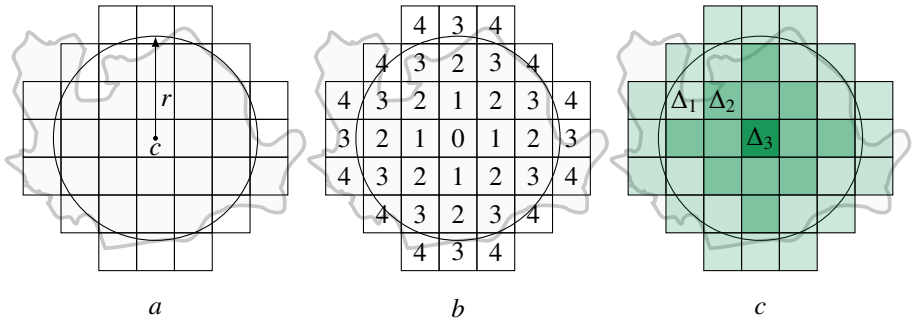


Figure 2: An example of the area coverage algorithm applied to a point of interest of interest. After defining the grid (a), the Manhattan distance from the center region is computed (b). Finally, each region is assigned to the right set (c). In this case, the maximum distance (σ) is 4, so we assign the regions belonging to the distance classes 4 and 3 to Δ_1, those belonging to the classes 2 and 1 to Δ_2 and the sole region belonging to the 0 class to the set Δ_3.

known as the *graticule* (Figure 1).

Longitude (`lng`) and latitude (`lat`) can be stored and represented according to several formats. In the following we use the *decimal degrees plus/minus* format, where latitude is positive if it is north of the equator (negative otherwise), and longitude is positive if it is east of the prime meridian (negative otherwise); for instance, 31.456764°(lat) and −85.887734°(lng) are two possible values.

Using a fixed precision in longitude and latitude (that is, choosing a fixed number of decimal points for their values) allows us to easily divide the planet's surface into a discrete grid. Since meridians get closer as they converge the poles, as can be seen in Figure 1, the portions of the Earth's surface defined by such a grid have varying areas depending on their position (Table 1). As the Earth is a *Geoid*, the shapes deriving from this grid are not exactly rectangles. However, for small areas, a planar or flat surface for Earth is still sufficient, as the local topography is far more significant than the curvature. The construction proposed in the following is not dependent on the size or shape of the regions. However, for simplicity and consistency in the figures, we represent the regions as squares.

In real applications, the precision in decimal points

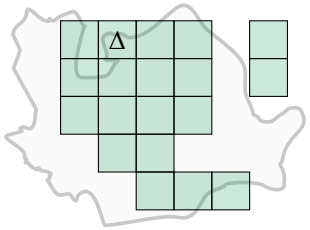


Figure 3: A sample area covered by an arbitrary grid.

for longitude and latitude should reflect the expected error of the device or sensor used for learning the location information. The precision and accuracy of mobile devices in determining their geographic position were proved to vary considerably depending on the context (urban areas, rural areas, etc.) [44].

In a detailed experiment on the accuracy of GPS sensors installed on mobile devices, Blum et al. show that the location is reported with a precision varying from 10 to 60 meters, depending on the device orientation and type, and, in cities, on the surrounding buildings [3]. Hence, when designing a system based on mobile devices it would reasonable to consider regions with sides tens of meters long.

For the purpose of this work we choose to consider the grid defined by longitude and latitude values with a precision of three decimal point places. This grid divides Earth's surface in a number of regions. We define the set of all regions as follows.

Definition 2. *We define $\mathcal{E}$ as the set of all regions in which Earth's surface is divided by the grid defined by the circles (called parallels) of latitude distant multiples of* 0.001° *from the equator and the arcs (called meridians) of longitude distant multiples of* 0.001° *from the Prime Meridian.*

The sides (in meters) of a region of side 0.001 degrees in terms of longitude and latitude vary depending on its position on the globe. Table 1 contains some reference values.

3.1. Areas and Points of Interest (AoI & PoI)

The purpose of this paper is to present a method able to preserve both user's and provider's privacy in location-aware applications. We imagine a scenario in

which the provider of such an application wants to be notified of the presence of the user in one of a predefined set of areas of interest (AoI). The areas of interest are selected by the provider, and each is composed of an arbitrary number of regions in $\mathcal{E}$, defined above. An area of interest may, for instance, represent a sensitive or interesting location for the purposes of the application. A number of concentric AoI around a point of interest (PoI) can be used to detect the user's vicinity to the PoI. A point of interest is a point on Earth's surface (at a specific latitude and longitude) whose position is deemed as significant by the provider; the point lies in one region in $\mathcal{E}$, and therefore, in the following, we identify a point of interest with the region containing it. In the following we present two approaches for selecting the regions of $\mathcal{E}$ in order to compose an area of interest, based on the provider's goals. Although both of them are used in this paper, for example purposes, as strategies to select the areas of interest by the service provider, we stress here that our construction is independent of the strategy used, and therefore can accommodate any other set selection mechanism. The first strategy assumes that the provider wants to select an arbitrarily shaped area of interest, and follows naturally from the idea of detecting the presence/absence of a person in this given zone. In order to do that, the provider of the service defines an area of interest by selecting a subset of $\mathcal{E}$ (Figure 3). The regions in the AoI need not to be contiguous, and there is no limitations in shape or size of the AoI. The set containing all of these regions is defined as Δ. A second approach is instead to monitor the user by detecting his proximity to a PoI as he approaches it. We achieve this goal without knowing the user's exact location by defining several concentric areas of interest around the PoI to be monitored. In the example shown in Figure 2 we use three AoI for this purpose, but this parameter can take any value deemed useful.

Let c be the PoI (having coordinates lng_c, lat_c) and let r be the range we are interested to monitor users around the center itself. First of all we choose a region such that it is the element of $\mathcal{E}$ that contains the point c. Then a number of adjacent elements (all belonging to $\mathcal{E}$) are added in order to form a grid, until the circle of center c and radius r is completely included in the grid, as shown in Figure 2a. Now let us label each region with its distance from the center region, using the standard *Manhattan distance* (Figure 2b). Assume that σ is the maximum distance value in the generated grid; we need to discuss two cases. If $(\sigma + 1) \mod 3 = 0$, we assign to the set Δ_3 each region labeled from 0 to $q - 1$, where $q = (\sigma + 1)/3$. Similarly, we fill the set Δ_2 with each square labeled from q to $2q - 1$ and the set Δ_1 with each square labeled from $2q$ to σ. If 3 does not divide $\sigma + 1$ exactly (i.e. $(\sigma + 1) \mod 3 \neq 0$) some rounding is required; we could for instance assign the first remaining class to Δ_1 and the second optionally remaining class to Δ_2 (Figure 2c). In that case, given $q = \lfloor(\sigma + 1)/3\rfloor$, the procedure can be formalized assigning each region labeled from 0 to $q - 1$ to the set Δ_3, each region labeled from q to $2q$ to the set Δ_2 and each region labeled from $2q + 1$ to σ to the set Δ_1. A generalization of this procedure for an arbitrary number of sets is formalized in Algorithm 2.

Algorithm 2: Area coverage (for d sets).

Input: c, r, $\mathcal{E}$, d;
Output: $\bar{S} = \Delta_1 \cup \Delta_2 \cup \cdots \cup \Delta_d$;

// Grid generation: define a grid $\bar{S}$ composed of contiguous elements of $\mathcal{E}$ that completely covers the circle of center c and radius r (Figure 2a).
$\bar{S} \leftarrow \emptyset$;
Define the circle C_r of center c and radius r;
Find δ_c, the element in $\mathcal{E}$ which contains c;
$\bar{S} \leftarrow \bar{S} \cup \{\delta_c\}$;
Starting from those elements of $\mathcal{E}$ contiguous to δ_c, insert in $\bar{S}$ each element of $\mathcal{E}$ completely or partially covered by C_r;
// Distance evaluation: for each $\delta \in \bar{S}$, compute the Manhattan distance from the central element (Figure 2b).
Assign to the element δ_c the label 0;
For each $\delta \in \bar{S}$, compute the Manhattan distance from δ_c and assign the result to δ as label;
Let σ be the greatest computed distance;
// Area definition: pack each $\delta \in \bar{S}$ in d concentric areas, equally assigning the number of labels to each area (Figure 2c).
Partition $\bar{S}$ in $\sigma + 1$ subsets $\bar{S}_{[i]}$ such that $\bar{S}_{[i]}$ contains the regions labeled with i;
$q \leftarrow \lfloor(\sigma + 1)/d\rfloor$;
$m \leftarrow (\sigma + 1) \mod d$;
for $j \leftarrow 1$ **to** d **do**
    if $m \neq 0$ **then**
        $\Delta_j \leftarrow \bar{S}_{[\sigma-q,\sigma]}$;
        $\sigma \leftarrow \sigma - (q + 1)$;
        $m \leftarrow m - 1$;
    else
        $\Delta_j \leftarrow \bar{S}_{[\sigma-q+1,\sigma]}$;
        $\sigma \leftarrow \sigma - q$;
    end
end
return $\Delta_1, \Delta_2, \ldots, \Delta_d$;

4. Spatial Bloom Filter

After defining a spatial representation $\mathcal{E}$ of Earth's surface and providing a way to identify geographical areas (and points) as elements of a subset of $\mathcal{E}$, we can use a set-based data structure like the Bloom filter to encode this information. However, the original definition of BF proves to be quite inefficient for this task, as it would be possible to encode only one area for each BF.

In the following we define a novel data structure called *Spatial Bloom Filter*. A spatial Bloom filter can be used, likewise the original BF, to perform membership queries on the originating set of elements without knowledge of the set itself. Contrary to the BF, however, a spatial Bloom filter can be constructed over multiple sets, and querying a spatial Bloom filter for an element returns the identifier of the specific set among all the originating sets in which the element is contained, minus a false positive probability (of assigning the element to the wrong set). Similarly to a classical BF, there is also a false positive probability that querying a SBF with an element outside the originating sets returns a positive result (wrongly assigning the element to one of the originating sets).

Before presenting the SBF formally and in order to better understand its construction, it is meaningful to denote that it holds an insightful property concerning false positives. Specifically, the probability of false positives, that is, the probability that an element is wrongly recognized as belonging to a specific originating set, depends on the order in which the sets have been encoded in the filter: a false positive can occur either when an element outside the originating sets is recognized as being part of one, or when an element that is part of an originating set is recognized as being belonging to a different one (sets are disjoint). The latter case, however, can only happen if the wrongly recognized set has been encoded later than the actual originating set.

This fundamental property allows to define an order of priority for the different originating sets, thus reducing the error probability for elements (areas) deemed more important. Considering the strategies described in the previous section for selecting areas of interests, this property is particularly useful when using SBFs to store location information. In the example presented in Section 3.1, for instance, we used a set of three different areas $S = \{\Delta_1, \Delta_2, \Delta_3\}$. Assuming the provider would prefer a more accurate monitoring of the area's central region, we assigned the highest label value (3) to the inner area. In the following we generally consider the sets as already ordered by priority, meaning that set Δ_2 is considered as having higher priority than Δ_1.

Definition 3. *Let $S = \{\Delta_1, \Delta_2, \ldots, \Delta_s\}$ be a set of areas of interest such that $\Delta_i \subseteq \mathcal{E}$ and S is a partition of the union set $\bar{S} = \bigcup_{\Delta_i \in S} \Delta_i$. Let O be the strict total order over S for which $\Delta_i < \Delta_j$ for $i < j$. Let also $H = \{h_1, \ldots, h_k\}$ be a set of k hash functions such that each $h_i \in H : \{0,1\}^* \to \{1, \ldots, m\}$, that is, each hash function in H takes binary strings as input and outputs a number uniformly chosen in $\{1, \ldots, m\}$. We define the* Spatial Bloom Filter *(SBF) over (S, O) as the set of pairs*

$$B^{\#}(S,O) = \bigcup_{i \in I} \langle i, \max L_i \rangle \ , \tag{9}$$

where I is the set of all values output by hash functions in H for elements of $\bar{S}$

$$I = \bigcup_{\delta \in \bar{S}, h \in H} h(\delta) \ , \tag{10}$$

and L_i is the set of labels l such that:

$$L_i = \{l \mid \exists \delta \in \Delta_l, \exists h \in H : h(\delta) = i\} \ . \tag{11}$$

A spatial Bloom filter $B^{\#}(S,O)$ can be represented as a vector $b^{\#}$ composed of m values, where the i-th value

$$b^{\#}[i] = \begin{cases} l & \text{if } \langle i, l \rangle \in B^{\#}(S,O) \\ 0 & \text{if } \langle i, l \rangle \notin B^{\#}(S,O) \end{cases} \ . \tag{12}$$

In the following, when referring to a SBF, we refer to its vector representation $b^{\#}$.

A SBF is built as follows. Initially all values in $b^{\#}$ are set to 0. Then, for each element $\delta \in \Delta_1$ and for each $h \in H$ we calculate $h(\delta) = i$, and set the i-th value of $b^{\#}$ to 1 (that is, to the label of Δ_1). We do the same for the elements belonging to the set Δ_2, setting $b^{\#}[i]$ to 2. We proceed incrementally until all sets in S have been encoded in $b^{\#}$. We observe that, following Definition 3, should a collision occur, the label with higher value is the one stored at the end of the process. Thus, values in the filter corresponding the elements in Δ_s will never be overwritten. This procedure is formalized in Algorithm 3 and depicted in Figure 4.

The verification process shall check whether an element δ_u is contained in a set $\Delta_i \in S$. Hence we verify whether $\delta_u \in \Delta_i$ if

$$\exists h \in H : b^{\#}[h(\delta_u)] = i \quad \text{and} \quad \forall h \in H, b^{\#}[h(\delta_u)] \geq i \ . \tag{13}$$

The procedure is described in Algorithm 4.

In practice, if any value of $b^{\#}$ in a position that corresponds to the output of one of the hash functions for δ_u is 0, then $\delta_u \notin \bar{S}$. If all the hashes map to elements of value i, then $\delta_u \in \Delta_i$ minus a false positive probability

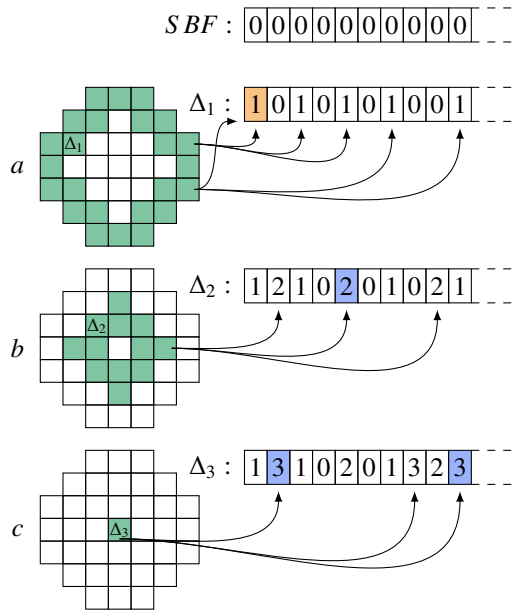


Figure 4: Areas Δ_1, Δ_2 and Δ_3 are used to construct a SBF. Three hash functions are used to map each element into the filter. Only the first ten elements of the SBF are shown. In *a*, two elements belonging to Δ_1 are processed by the hash functions, resulting in six 1 value elements to be written into the SBF. The first element collides as highlighted. This kind of collision is the same that may occur in a classic Bloom Filter. After each element in Δ_1 is processed, the algorithm processes elements in Δ_2 (*b*) and finally in Δ_3 (*c*). Note that the collisions in *b* and *c* are different from the previous one and are SBF specific. Areas marked with a greater label are assumed to be more important from the provider point of view and overwrite elements of lower value on collision.

which is discussed in the following. The same applies if at least one hash maps to an element of value i and the remaining hashes map to elements of value $> i$. In fact, since when a collision occurs the highest value is stored, a lower value could be overwritten.

Similarly to the case of the original Bloom filter (Section 2.1), a false positive probability p exists when determining whether an element belongs to the set $\bar{S}$ or not. In the case of a spatial Bloom filter $B^{\#}(S, O)$, however, the probability p can be split into several probabilities p_i, each one subset-specific. Specifically, p_i is the probability that an element δ is wrongly recognized as belonging to the set Δ_i, while either $\delta \notin \bar{S}$ or $\delta \in \Delta_j$, with $j < i$. For instance, a false positive assigned to the set Δ_s occurs if each hash collides with a value s in $b^{\#}$. As stated in (4), for a classical BF the false positive probability can be approximated as

$$\left(1 - e^{-\frac{kn}{m}}\right)^k \quad . \tag{14}$$

Thus, adjusting the exponent with the subset-specific

Algorithm 3: Spatial Bloom Filter construction.

Input: $\Delta_1, \Delta_2, \ldots, \Delta_s, H$;
Output: $b^{\#}$;

```
for i ← 1 to s do
    foreach δ ∈ Δ_i do
        foreach h ∈ H do
            b#[h(δ)] ← i;
        end
    end
end
return b#;
```

Algorithm 4: Spatial Bloom Filter verification.

Input: $b^{\#}, H, \delta_u, s$;
Output: Δ_i;

```
i = s;
foreach h ∈ H do
    if b#[h(δ_u)] = 0 then
        return false;
    else
        if b#[h(δ_u)] < i then
            i ← b#[h(δ_u)];
        end
    end
end
return Δ_i;
```

number of elements, we can denote this probability as follows:

$$p_s \approx \left(1 - e^{-\frac{k|\Delta_s|}{m}}\right)^k \quad . \tag{15}$$

Similarly, we can compute the probability to wrongly assign an element to the set Δ_{s-1} considering all of the possible collisions with elements belonging to Δ_s and Δ_{s-1}, excluding those deriving from collisions with elements belonging to Δ_s entirely. Hence

$$p_{s-1} \approx \left(1 - e^{-\frac{k|\Delta_s \cup \Delta_{s-1}|}{m}}\right)^k - p_s \quad . \tag{16}$$

We can proceed likewise to the last set:

$$p_1 \approx \left(1 - e^{-\frac{k|\bar{S}|}{m}}\right)^k - p_s - p_{s-1} - \cdots - p_2 \quad . \tag{17}$$

It follows that $p_1 + p_2 + \cdots + p_s = p$, where p is the same false positive probability provided in (4) if $|\bar{S}| = n$.

In the following we assume that the possibility of false positives among sets (that is, having elements in $\bar{S}$ assigned to the wrong set) is deemed as generally acceptable when using a SBF.

Let us finally note that a SBF bears some resemblance to a bloomier filter [8, 7], a variant of the classical Bloom filter used for storing binary functions instead of sets. We could in fact define the originating sets through a function, and build the corresponding bloomier filter. However, in the case of a spatial Bloom filter we have an error probability between different Δ's, but we know exactly whether a $\delta \in S$ or not. A bloomier filter, instead, would behave in the opposite way: the function always outputs the correct Δ, but there exists a probability that a $\delta \notin S$ will be wrongly recognized as belonging to one Δ. Considering location-aware applications, we deem an error in positioning over two contiguous areas of interest as acceptable, while mistakenly recognizing a position outside the areas of interests (even by far) as inside as much more problematic. Therefore, we believe that the proposed spatial Bloom filters are better suited to be used in the location-aware context, while bloomier filters might still be useful in specific application scenarios.

5. Private Positioning Protocols

A major feature of SBFs is that they allow private computation of location based information. We show this by providing two protocols based on spatial Bloom filters that address the problem of location privacy in a location-aware application. In general, a location-aware application is any service that is based on (partial) knowledge of the geographic position of the user. In this work, however, we focus on applications in which the service provider has an interest in learning when the user is within an area (or close to a point) of interest. The security discussion in the following is based on the privacy definition we provide in the preliminaries (Section 2).

The protocols we present are designed for a secure multi-party computation setting, where the user and the service provider are mutually distrusting, and therefore do not want to disclose private information to the other party. In the case of the user, private information is his exact location. The service provider, instead, does not want to disclose the monitored areas. We address this problem by providing a scheme that allows the provider of a service to detect when the user is within an area of interest, without requiring the user to reveal his exact position to the provider. At the same time, the privacy of the provider is also guaranteed with respect to the areas of interest. The privacy benefits for the user are double: first and foremost, the relative location is only revealed when the user is within predetermined areas, and remains private otherwise. Secondly, even when presence in an area is detected, only this generic information is learned by the provider, and not the actual position. Following the area coverage mechanism proposed in Section 3.1, for instance, the provider learns the distance from the central area to a certain extent, while the direction from which the user approaches it stays private. Dividing the area around the point of interest in a different manner may reveal instead the direction but conceal the distance within the area range. Moreover, should the provider decide maliciously to monitor a wide zone (such as a nation or a large urban area) by considering each included region in $\mathcal{E}$ as a single area, the filter would increase significantly in size: this would be immediately evident to the user (beside becoming unpractical due to the abnormal size). Therefore, a simple sanity check on the size of the filter will effectively prevent this deviation from the protocol.

In the following we discuss two different settings: in the first setting the user communicates directly with the service provider, who computed beforehand a spatial Bloom filter relative to the areas he is interested in monitoring. In the second setting, instead, the service provider computes the SBF, but communication with the user is handled by a third party, to which the provider outsources the task. In both setting, no trust is implied among the parties, including the third party, and we assume the parties do not collude with each other. We work in the honest-but-curious setting, as defined in the preliminaries (Section 2).

5.1. Two-party Scenario

In the two-party scenario the communication happens between the service provider *Paul* and the user *Ursula*. We assume the user has access to a positioning system that allows her to determine her geographic position. Ursula is interested in using a location-aware service provided by Paul, but she does not want to disclose her exact position. Paul, on the other hand, wants to learn if Ursula is close to some points of interest or is within an area of interest, but he does not want to share with her these locations. Since the two parties are mutually distrusting, this is a secure multi-party computation problem.

We propose Protocol 1, that addresses the problem securely by disclosing only the identifier i of the area Δ_i in which the user is. Intuitively, the protocol works as follows. Paul creates a SBF for the points and areas of interest as described in the previous sections. He encrypts the filter (by encrypting each value therein) with an encryption scheme that allows the private Hadamard product defined in Algorithm 1, and sends it to Ursula.

Protocol 1: Two-party private positioning protocol between service provider Paul and user Ursula.

Before any communication, the provider selects the areas of interest $\Delta_1, \ldots, \Delta_s \subset \mathcal{E}$. Then, he selects the desired false positive probability p, and determines k and m according to (5) and (6) respectively. Finally, following the notation of Definition 3, the provider computes the spatial Bloom filter $b^{\#}$ over $\bar{S}$ using Algorithm 3.

1. The service provider Paul generates a public and private key pair using a multiplicative homomorphic encryption scheme, and sends the public key to the user Ursula.
2. Paul sends to Ursula the encryption of the precomputed SBF $\text{Enc}\left(b^{\#}\right)$, the set of k hash functions H, the value m and the conventional grid $\mathcal{E}$.
3. At regular time intervals, or when required by the specific application, Ursula determines her geographic position and selects the corresponding grid region $e_u \in \mathcal{E}$. Then, following Algorithm 3 and using the values and functions shared by Paul, she builds a spatial Bloom filter $b_u^{\#}$ over $\{e_u\}$ and counts the number z of values equal to 1 therein.
4. Ursula computes $e^{\#} = \text{Enc}\left(b^{\#}\right) \boxdot b_u^{\#}$ using the homomorphic properties of the encryption scheme (Algorithm 1). Then she applies a random permutation to the values in the filter, and sends z and the result to Paul.
5. Paul decrypts $e^{\#}$ and counts all non-zero values. If the resulting number is $< z$, Ursula's position is outside of the areas on which the SBF was built. Otherwise, the value i, corresponding to area Δ_i identifying Ursula's position (minus error probability p_i), is the smallest non-zero value in $\text{Dec}\left(e^{\#}\right)$.

Ursula creates a SBF for the set composed only of her position in the grid. The filter is binary, since 0's and 1's are the only possible values in a filter with only one point of interest. Then Ursula computes the entrywise homomorphic product of the received SBF with the one she just computed: this way, only the values of the encrypted filter corresponding to a 1 in her filter are preserved, while the others take value 0. Then she shuffles the values in the resulting encrypted filter and sends the randomly ordered filter back to Paul.

Security Definition. In a two-party setting implementing Protocol 1, the computation is achieved privately if at the end of the protocol execution Paul learns only $i \in \{1, \ldots, s\}$, and Ursula learns nothing.

In the following we analyze the security of the protocol with respect to the above definition. In order to quantify the information learned by Paul during the protocol execution, we introduce an arbitrarily small security parameter $\mathcal{E}$. Then, we prove that the probability of Paul learning useful information is upper-bounded by the chosen $\mathcal{E}$.

Security Analysis. As stated in the security definition, a successful execution of Protocol 1 should guarantee three conditions: correctness of the result for Paul, privacy for Ursula's position and privacy of the areas encoded in the filter by Paul. We discuss the three conditions in the following.

The protocol ends correctly if the number of non-zero values read in the decrypted $e^{\#}$ by Paul is $< z$ in case Ursula is outside the areas of interests; in case Ursula is within an area, the protocol ends correctly if the number of non-zero values is equal to z, and the area is identified by the smallest non-zero value, minus error probability p_i. The former case is always true, for the properties of Definition 3, as explained in Section 4. In the latter case, the false positive probability p_i for each area i is determined by Paul according to (17) during filter creation. It is therefore Paul himself who decides the correctness bounds of the protocol.

The second condition (Ursula's privacy) is respected if Paul learns only in which (predefined) area the user is, and not her exact position at the end of the protocol. If the user is outside the areas of interest, the provider should learn nothing. Ursula encodes her position in $b_u^{\#}$ at step 3 of the protocol, and sends the encrypted filter $e^{\#} = \text{Enc}\left(b^{\#}\right) \boxdot b_u^{\#}$ back to Paul after performing a random permutation on the order of its values. The homomorphic properties of a public key encryption scheme guarantee that Paul can only learn a number of values from $b^{\#}$ that corresponds to non-zero values in $b_u^{\#}$ [21]. At the same time, the random permutation prevents him from understanding to which position in $b^{\#}$ each of these values corresponds to, therefore making it impossible to reconstruct Ursula's filter based on the order of elements. If the number of non-zero values is z, and all take the value i corresponding to an area of interest, Paul only learns the area of interest. In case, instead, some values are $> i$ for some of the positions on the grid within the area of interest, then Paul learns the area of interest Δ_i and a pattern of values. The same applies in case Ursula is outside of any area of interest, but the decryption of $e^{\#}$ reveals a number of non-zero values $w < z$. In the following we focus on the latter scenario, as a potential attack exploiting the pattern information could reveal the user's position even when she is outside

the areas of interests. In fact, if the pattern is unique for a position on the grid, Paul may be able to learn Ursula's position by performing an exhaustive search on all the possible positions on the grid: given the irreversibility of (spatial) Bloom filters, the complexity of the attack is linear to the number of such positions. We prevent this attack by having each pattern shared by at least a possible positions: in which case we achieve a-anonymity for the user's position even in case of an exhaustive search. We define an arbitrarily small security parameter ϵ, and we consider the privacy condition to be met if the probability of Paul learning Ursula's position is $\frac{1}{a} < \epsilon$. For each number $w \in \{1, \dots, z\}$ of non-zero values obtained by Paul, we can estimate the value of a based on the number of possible positions in $\mathcal{E}$ and the number of areas of interest s. In particular, we calculate the number of possible patterns for a given w as the combinations with repetitions of length w, $\binom{s+w-1}{w}$. Based on this, we can estimate the average value $\bar{a}$ for the different a's of all possible combination with repetitions to be

$$\bar{a} = \frac{|\mathcal{E}|}{\sum_{w=1}^{k} \binom{s+w-1}{w} + 1} \quad , \tag{18}$$

if we assume a linear distribution of the values $\{1, \dots, s\}$ over the filter. The security condition is hence met if $\frac{1}{a} < \epsilon$ for all a's relative to any possible w. We note, from the formula above, that this mostly depends on the number of areas of interest s and, on a lesser extent, on the number of hashes k (since $z \leq k$). These two values can therefore be tuned in order to achieve the desired security parameter ϵ, as both values are selected before the creation of the filter. Considering the order of magnitude of $|\mathcal{E}|$, which is 10^{12}, an appropriately built filter can satisfy a security parameter $\epsilon = 10^{-6}$ for most values of k and s. Thanks to the fine grained nature of the grid, even geographically limited settings which restricts the area of potential positions of the user can achieve reasonable security margins ($\epsilon \approx 10^{-3}$): in fact, small areas of a few square kilometers already include several millions possible positions (Section 3).

Finally, the privacy of the service provider, that is, the secrecy of the areas encoded in the filter, is ensured by the encryption of the filter itself. Ursula, in fact, never learns the cleartext of the filter, as she is able to perform the multiplication of step 4 in the encrypted domain thanks to the homomorphic properties of the public key encryption scheme.

Protocol 2: Three-party private positioning protocol among provider Paul, third party Olga and user Ursula.

Before any communication, the provider selects the areas of interest and creates the corresponding spatial Bloom filter similarly to Protocol 1.

1. The service provider Paul generates a public and private key pair using a multiplicative homomorphic encryption scheme, and sends the public key to the third party Olga.
2. Paul sends to Olga the encryption of the precomputed spatial Bloom filter $\text{Enc}\left(b^{\#}\right)$ and the value m. Then, Paul sends to the user Ursula the set of k hash functions H and the conventional grid $\mathcal{E}$.
3. At regular time intervals, or when required by the specific application, Ursula determines her geographic position and selects the corresponding grid region $e_u \in \mathcal{E}$. Then, she computes the values $\{v_1, \dots, v_k\}$ where $v_i = h_i(e_u)$, and sends them to Olga.
4. Olga receives the values from Ursula and builds $b_o^{\#}$, by assigning $b_o^{\#}[v_i] = 1$ for every $v_i \in \{v_1, \dots, v_k\}$. Then, she calculates z as the number of 1's in $b_o^{\#}$.
5. Olga computes $e^{\#} = \text{Enc}\left(b^{\#}\right) \boxdot b_o^{\#}$ using the homomorphic properties of the encryption scheme (Algorithm 1). Then she applies a random permutation to the values in the filter, and sends z and the result to Paul.
6. Paul decrypts $e^{\#}$ and counts all non-zero values. If the resulting number is $< z$, Ursula's position is outside of the areas on which the SBF was built. Otherwise, the value i, corresponding to Ursula's area Δ_i (minus error probability p_i), is the smallest non-zero value in $\text{Dec}\left(e^{\#}\right)$.

5.2. *Three-party Scenario*

In the three-party scenario the communication does not happen directly between the service provider and the user (Protocol 2). The service provider is responsible for creating and managing the filter, but the verification of user values and therefore all direct communication with the user is outsourced to a third party, whom we call *Olga*. We introduce the third party in order to decrease the computation and communication burden imposed on the user Ursula. In fact, while it is reasonable to assume that the service provider has adequate resources in terms of computational power and bandwidth to manage filters of big size, the same assumption can not be made for the user, who might be constrained to the limited resources of a mobile device such as a smartphone. Therefore, we offload all onerous tasks to the provider and the third party, who is also assumed to

be communication and computationally capable.

Security Definition. In a three-party setting implementing Protocol 2, assuming that no information other than the one implied by the protocol is shared between the parties (parties do not collude), the computation is achieved privately if at the end of the protocol execution Paul learns only $i \in \{0, \ldots, s\}$, while Olga and Ursula learn nothing.

Security Analysis. The security of the three-party protocol follows that of the two-party protocol above. The introduction of the third party means however that the user sends her unencoded hash values to the third party, who performs the private Hadamard product. This exposes the user to an attack on the spatial Bloom filter by the third party. While Bloom filters have proved to be irreversible, an exhaustive search may reveal to Olga the input used to produce the received hash outputs. This attack, however, assumes knowledge of $\mathcal{E}$ by Olga. The conventional grid $\mathcal{E}$ represents in fact the coding scheme (or ordering) of the elements on the geographical grid: that is, which value is to be given as input to the hash functions for each position. Since this information is not required by Olga for the execution of the protocol, the user and the provider can agree on an encoding scheme (which can simply be a random ordering of the geographical grid elements) unknown to the third party, thus preventing her from running a search attack. We note that the same goal can also be achieved by using keyed hash functions, which would however require a key exchange between the two parties.

A second threat to which the user is exposed is due to the deterministic nature of the hash results for the same input. In fact, the third party may easily know if the user is revisiting the same grid position twice by comparing the hash digests. In settings in which this is considered unacceptable, a temporal-based variation of the above encoding of the geographical grid can be used.

6. Evaluation

In this section we provide an in depth evaluation of several SBF properties. First, we analyze the computation and communication overhead of the solution. Then, we analyze how the probability of false positives changes for different values of k (the number of hash functions). Finally, we discuss the density of the filter, which has significant implications on the value of the security parameter ϵ.

The computational complexity for the insertion and the verification of a single element in a SBF is linear in the number k of hash functions used for the filter. The private Hadamard product has instead a computational cost linear to the length of the filter m.

In the following we provide an estimation of the communication overhead, and we evaluate the computational cost required for an execution of the protocol (Table 2). While being a generally compact data structure, a SBF built over a significantly large number of sets can require a sizable amount of memory. While m bits are needed to store a classical Bloom filter b, a SBF needs more bits due to the labeling of subsets Δ. More precisely, in order to store $b^{\#}$, $(\lfloor \log_2 s \rfloor + 1)\, m$ bits are needed. Depending on the number of areas and the desired error probability, a SBF could require a storage space (and communication cost when transmitted) not suitable for constrained scenarios, as in the case of mobile devices: in this case, the use of the second protocol, involving a third party, can significantly reduce both the communication and the computational load on the user's device. For instance, let us consider hash functions with a 16-bit digest (i.e. $m = 2^{16}$) and an area of interest divided into six sub areas. Since $s = 6$, a SBF built on these functions needs $(\lfloor \log_2 6 \rfloor + 1)\, 2^{16}$ bits, resulting in a data structure of approximately 24 KB. Using the second protocol, the communication is reduced to just 160 bits, assuming a number of hash functions $k = 10$. The computationally intensive operations to be performed by each party, and the communication costs are summarized for both protocols in Table 2. These results indicate that SBF's can be adopted in most circumstances and real-world scenarios, either with direct com-

	Brussels city			**Belgium**		
Region extension	161.28 Km^2			30 536 Km^2		
δ^{50th} **per region**	20 958			3 965 715		
Monitored areas	15			1023		
Area extension	0.25 Km^2			1 Km^2		
δ^{50th} **per area**	33			130		
Monitored δ^{50th}	495 (2.4%)			132 990 (3.3%)		
Hashes *(false positives)*	2 - 30			2 - 70		
	SBF1	**SBF2**	**SBF3**	**SBF1**	**SBF2**	**SBF3**
Hashes *(security analysis)*	10	10	10	10	10	10
Hash mapping	2^{13}	2^{14}	2^{15}	2^{21}	2^{22}	2^{23}
Binary SBF size	4 KB	8 KB	16 KB	2.5 MB	5 MB	10 MB

Table 3: SBF evaluation examples.

	User	Provider	Third party
Comp. (2-p)	1 SBF-insertion, 1 Private Hadamard Product	1 decryption, 1 match count	
Comp. (3-p)	k hashes	1 decryption, 1 match count	1 SBF-completion, 1 Private Hadamard Product
Comm. (2-p)	$O(m)$ $(\lfloor \log_2 s \rfloor + 1)\, m$	$O(m)$ $(\lfloor \log_2 s \rfloor + 1)\, m$	
Comm. (3-p)	$O(\log_2 m)$ $k\,(\lfloor \log_2 m \rfloor + 1)$	$O(m)$ $(\lfloor \log_2 s \rfloor + 1)\, m$	$O(m)$ $k\,(\lfloor \log_2 m \rfloor + 1) + (\lfloor \log_2 s \rfloor + 1)\, m$

Table 2: Computation and communication load for stakeholders.

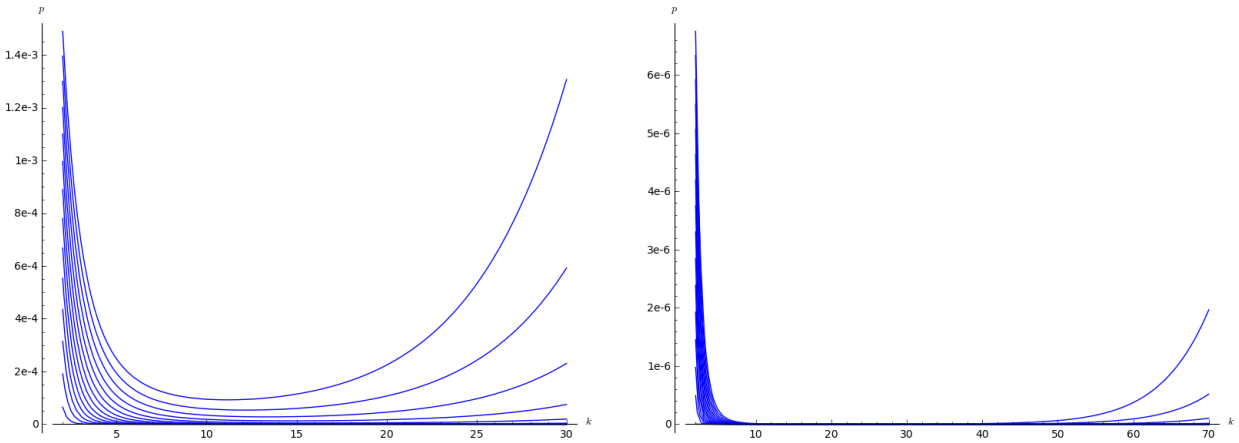
munication between the user and the service provider, or through the use of a third party.

In order to discuss the security properties and the false positive probability, we introduce two examples based on actual geographical regions: the metropolitan area of the city of Brussels, first, and the whole country of Belgium, second. These two examples let us provide insight in the usefulness of the proposed solution in real-world settings, and for realistically sized regions. In order to perform the tests, we implemented a prototype, but fully functional version of the filter creation and query routines. Using that, we construct a set of test filters of different sizes for the different parameters. The details of the two experiments are summarized in Table 3. As the mentioned areas are located across the 50th parallel North, we approximate each element $\delta \in \mathcal{E}$ to a 70×110m rectangle (as described in Section 3 and Table 1). Such a rectangle covers an overall surface of 0.0077 Km^2. For the purpose of these examples we will refer to this kind of rectangles as to δ^{50th}. In the first example, the provider needs to monitor 15 critical areas within the Brussels city region. This region measures approximately 161 Km^2 and thus needs a total of 20958 δ^{50th} to be fully covered. We assume that the 15 areas which the provider monitors within this region are 0.25 Km^2 each, resulting in approximately 33 δ^{50th} needed for any single area. The second example focuses instead on a larger region, namely the country of Belgium. Its area is approximately 30000 Km^2. Here we assume 1023 monitored areas covering 1 Km^2 each. Therefore, around 130 δ^{50th} are needed in order to cover each of them.

We start the analysis of the filters by calculating the false probability function for the areas of interest encoded in the filters, with respect to the number k of hash functions used. For this calculation we use, in the case of the first example, an SBF with $m = 8192$ (that is, a total size of 4 KB) and we test the value of k starting from 2 hash functions and up to 30. For the second example we use instead a filter of 5MB of size and $m = 4194304$, which we plot for k between 2 and 70. Results are presented in Figure 5. As evident from the graphs, the optimal number of k lies in between 8 and 13 for the first case and between 9 and 40 for the second example. Therefore, we choose a value of $k = 10$ for our following analysis.

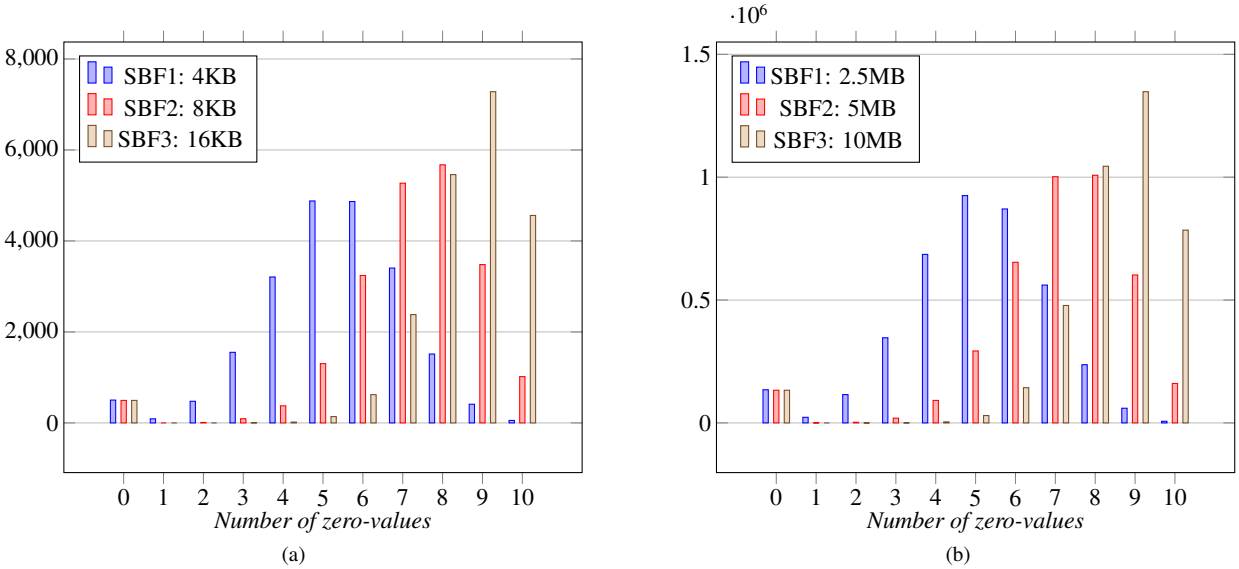
It is remarkable to note that the area considered more sensitive (the one labeled with the highest value following our assumption) almost always holds a smaller false positives probability (actually a really small one) with respect to the other regions. In general, $p_i < p_j$ for $i > j$. However, this property may not stand true when the filter is exceedingly dense: this condition is reached when the filter is too small in size or when the number of hashes is too high. This complex phenomenon surely deserves attention and could be further studied. For the purpose of this paper, it is sufficient to note that a SBF can be tuned through m and k in order to meet the given application requirements.

Having determined the value of k, we proceed to create a number of test filters of different sizes, in order to evaluate their density. We study the density of a filter by looking at the δ's outside the areas of interest. In a filter of infinite length, querying the filter over a $\delta \notin S$ should result in k zero values. In general, the shorter the filter, the more the occurrences in which a $\delta \notin S$ will result in a number of non-zero values being returned. In the chart shown in Figure 6, three filters of different sizes are built for each example, and the number of δ's resulting in exactly v_i non-zero values are plotted. Intuitively, the larger the filter, the lower the number of occurrences for high values of v_i. This has strong implications on the privacy properties of the filter: if we consider a $\delta \notin S$ resulting in 0 non-zero values, a person standing in that region will have (v_{10})-anonymity. The anonymity of a person in an area resulting in 1 non-



(a) (b)

Figure 5: False positives probability reported varying the number of hash functions following the examples provided in Table 3. Each plotted function represents the false positives probability for a specific area Δ_i. Figure 5a, related to the Brussels city region, reports functions for all of the 15 monitored areas while, for ease of reading, Figure 5b, related to Belgium as a whole, reports just a few functions among the existing 1023.



(a) (b)

Figure 6: Density of three sample filters used to run the examples provided in Table 3. For each SBF we show the distribution of zero-values among the mapped elements δ^{50th}. Figure 6a refers to the Brussels region while Figure 6b refers to Belgium as a whole.

zero value will have an anonymity of v_9 divided by the number of combinations with repetitions with $w = 1$ on average, as explained in Section 5, and so on. We conclude that the filters of larger sizes in the examples are providing effective anonymity in most cases, namely 85% and 58% for the first and second examples respectively, and a lower anonymity grade in the rest of the areas. By increasing the size of the filter to $2^3 = 8$ times that of the presented examples (S3), we reach in both cases effective anonymity for the filters in 99.999% of the cases. We can scale exponentially higher by simply increasing the size of the filter further. This is pictured in Figure 7, where the plotted function represents the probability of an area $\delta \notin S$ to show a unique pattern in the filter, and therefore being identifiable by a honest-but-curious provider. We consider patterns to be identifiable for any $v_{i\geq 2}$. A probability close to 0 denotes therefore a high privacy guarantee, while lower

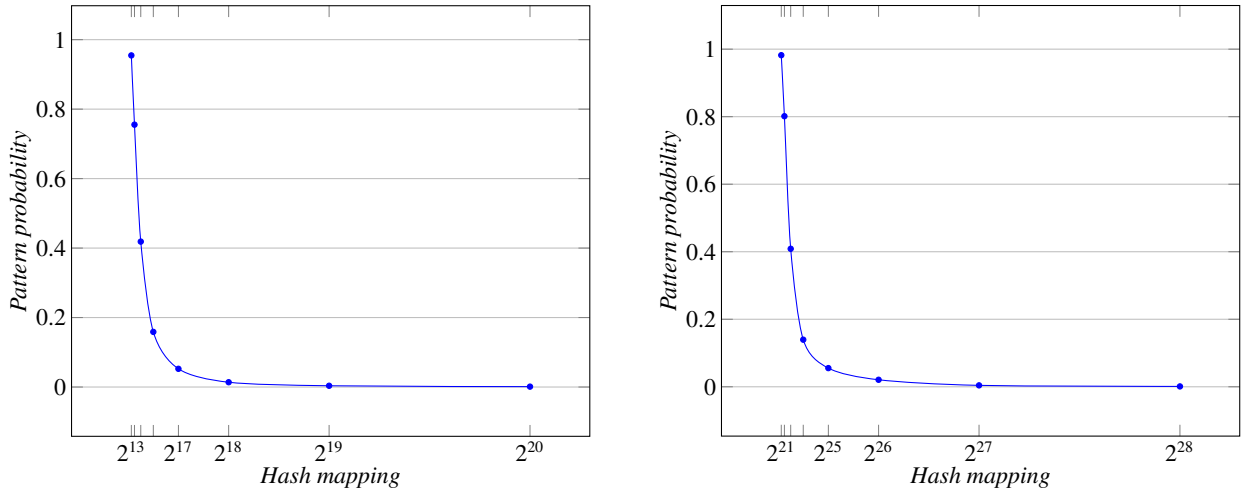


Figure 7: Probability of patterns that could lead to the identification of the position of the user when outside an area of interest by an honest-but-curious provider, with respect to the hash mapping m. Figure 7a is relative to Brussels, while Figure 7b to Belgium (see Table 3). The size of the filter is linear to the hash mapping, as per the communication cost formula in Table 2.

probabilities mean a weaker privacy property. As evident from the graphs, in the case of the Brussels area, any hash mapping $m \geq 2^{18}$ will provide a probability of identifying patterns close to 0, while in the case of Belgium this is true for values of $m \geq 2^{26}$.

7. Conclusions

In this paper we present a novel privacy-preserving primitive, the Spatial Bloom Filter (SBF). Based on the classical Bloom filter, the SBF extends it by allowing multiple different sets to be encoded in a single filter. Spatial Bloom filters are particularly suited to store location information, when such information is represented in a set-based format: in order to show this, we provide a spatial representation system for geographic areas, which allows us to encode positioning information (such as the one produced by GPS devices) into an SBF.

A main characteristic of spatial Bloom filters is to allow privacy-preserving location queries. In particular, we can encode into an SBF a list of sensitive areas and points located in a geographic region of arbitrary size. In many applications, such as law enforcement surveillance, military tracking or even location-based advertising, a service provider is interested in detecting the presence of a user within predetermined areas of interest, or his proximity to points of interest. In order to avoid constant tracking, however, the provider should be notified only when the current location of a user lies within those areas, and not otherwise. At the same time, the provider might have an interest in keeping the location of these sensitive areas hidden from the users (imagine, for instance, a tracking system for convicts sentenced to house arrest, or the surveillance of military bases). This is a typical secure multi-party computation problem: different players want to compute cooperatively the result of a function, but without disclosing their inputs to each other. Thanks to the properties of spatial Bloom filters, we can build a private location protocol that solves this problem.

In this paper we propose two privacy-preserving protocols for location-based services based on spatial Bloom filters. The protocols use the homomorphic properties of a public key encryption scheme (such as Paillier's cryptosystem) in order to guarantee both user's and provider's privacy: the provider only learns in which (predefined) area the user is, but not his exact position, and only if the user is within those areas; the user learns nothing. In the first protocol, the user communicates directly with the service provider. The second protocol, instead, provides an alternative for users unable to perform complex computational tasks (this might be the case, for instance, of embedded devices with limited capabilities). In the latter protocol most of the computation is outsourced to a third party, but without assuming any trust. In the paper we prove the security of both protocols, and we show the results of a test implementation, which allows us to establish the security margins of the construction. The results highlight the flexibility

of spatial Bloom filters: we can in fact satisfy any desired privacy bound by calibrating the parameters of the filter, as shown in Figures 5 and 7. We test the implementation using two real geographic areas: the city of Bruxelles, and Belgium. In the first case, the size of the filter can be as small as a few KB's, while the whole Belgium can be covered with just a few MB's. This results open the way to actual implementation of privacy-preserving protocols in location-aware applications, and address for the first time the problem of location privacy in the secure multi-party computation setting.

Finally, in the discussion of the bounds to the density of spatial Bloom filters we pointed out that, in some limited and specific cases, the most sensitive areas in the filter may not hold the smallest false positives probability when the filter is exceedingly dense. We suggest a study of this phenomenon as future work, investigating the mathematical properties of the SBF primitive and focusing on the study of the number of hashes, the dimension of the filter, the number of areas and, in general, on SBF tuning and optimization.

Acknowledgments

The authors would like to acknowledge Marco Miani for the code used in producing Figure 1.

References

[1] Avoine, G., Calderoni, L., Delvaux, J., Maio, D., Palmieri, P., 2014. Passengers information in public transport and privacy: Can anonymous tickets prevent tracking? Int J. Information Management 34, 682–688. doi:10.1016/j.ijinfomgt.2014.05.004.

[2] Bloom, B.H., 1970. Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13, 422–426.

[3] Blum, J.R., Greencorn, D.G., Cooperstock, J.R., 2012. Smartphone sensor reliability for augmented reality applications, in: MobiQuitous, Springer. pp. 127–138.

[4] Blumberg, A.J., Eckersly, P., 2009. On Locational Privacy, and How to Avoid Losing it Forever. https://www.eff.org/wp/locational-privacy.

[5] Bonomi, F., Mitzenmacher, M., Panigrahy, R., Singh, S., Varghese, G., 2006. An improved construction for counting bloom filters, in: Azar, Y., Erlebach, T. (Eds.), ESA, Springer. pp. 684–695.

[6] Calderoni, L., Maio, D., Palmieri, P., 2012. Location-aware mobile services for a smart city: Design, implementation and deployment. JTAER 7, 74–87.

[7] Charles, D.X., Chellapilla, K., 2008. Bloomier filters: A second look, in: Halperin, D., Mehlhorn, K. (Eds.), ESA, Springer. pp. 259–270.

[8] Chazelle, B., Kilian, J., Rubinfeld, R., Tal, A., 2004. The bloomier filter: an efficient data structure for static support lookup tables, in: SODA, SIAM. pp. 30–39.

[9] Chen, S., Xu, L., Chen, Z., 2007. Secure anonymous routing in trust and clustered wireless ad hoc networks, in: Communications and Networking in China, 2007. CHINACOM '07. Second International Conference on, pp. 994–998. doi:10.1109/CHINACOM.2007.4469552.

[10] Cramer, R., Damgård, I., Nielsen, J.B., 2012. Secure Multiparty Computation and Secret Sharing - An Information Theoretic Approach. Book Draft.

[11] Dawood, R., Yew, J., Jackson, S.J., 2010. Location aware applications to support mobile food vendors in the developing world, in: Mynatt, E.D., Schoner, D., Fitzpatrick, G., Hudson, S.E., Edwards, W.K., Rodden, T. (Eds.), CHI Extended Abstracts, ACM. pp. 3385–3390.

[12] Defrawy, K.M.E., Tsudik, G., 2011. Privacy-preserving location-based on-demand routing in manets. IEEE J. on Selected Areas in Communications 29, 1926–1934.

[13] Fan, L., Cao, P., Almeida, J.M., Broder, A.Z., 2000. Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. 8, 281–293.

[14] Geravand, S., Ahmadi, M., 2013. Bloom filter applications in network security: A state-of-the-art survey. Computer Networks 57, 4047–4064.

[15] Ghinita, G., Rughinis, R., 2013. A privacy-preserving location-based alert system, in: Knoblock, C.A., Schneider, M., Kröger, P., Krumm, J., Widmayer, P. (Eds.), SIGSPATIAL/GIS, ACM. pp. 422–425.

[16] Gruteser, M., Grunwald, D., 2003. Anonymous usage of location-based services through spatial and temporal cloaking, in: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, ACM, New York, NY, USA. pp. 31–42. URL: http://doi.acm.org/10.1145/1066116.1189037, doi:10.1145/1066116.1189037.

[17] Guo, D., Wu, J., Chen, H., Luo, X., 2006. Theory and network applications of dynamic bloom filters, in: INFOCOM, IEEE.

[18] Hawelka, B., Sitko, I., Beinat, E., Sobolevsky, S., Kazakopoulos, P., Ratti, C., 2013. Geo-located twitter as the proxy for global mobility patterns. CoRR abs/1311.0680.

[19] Jiazhu, D., Zhilong, L., 2013. A location authentication scheme based on proximity test of location tags, in: ICINS 2013, pp. 1–6. doi:10.1049/cp.2013.2449.

[20] Jung, T., Li, X.Y., 2012. Search me if you can: Privacy-preserving location query service. CoRR abs/1208.0107.

[21] Kikuchi, H., Sakuma, J., 2014. Bloom filter bootstrap: Privacy-preserving estimation of the size of an intersection. JIP 22, 388–400. doi:10.2197/ipsjjip.22.388.

[22] Kim, I.Y., Kim, K.C., 2008. A resource-efficient ip traceback technique for mobile ad-hoc networks based on time-tagged bloom filter, in: Convergence and Hybrid Information Technology, 2008. ICCIT '08. Third International Conference on, pp. 549–554. doi:10.1109/ICCIT.2008.292.

[23] Kirsch, A., Mitzenmacher, M., 2006. Distance-sensitive bloom filters, in: Raman, R., Stallmann, M.F. (Eds.), ALENEX, SIAM. pp. 41–50.

[24] Kulik, L., 2009. Privacy for real-time location-based services. SIGSPATIAL Special 1, 9–14.

[25] Li, N., Raj, M., Liu, D., Wright, M., Das, S.K., 2012. Using data mules to preserve source location privacy in wireless sensor networks, in: Bononi, L., Datta, A.K., Devismes, S., Misra, A. (Eds.), ICDCN, Springer. pp. 309–324.

[26] Lien, I.T., Lin, Y.H., Shieh, J.R., Wu, J.L., 2013. A novel privacy preserving location-based service protocol with secret circular shift for k-nn search. IEEE Transactions on Information Forensics and Security 8, 863–873.

[27] Liu, Y., Chen, X., Li, Z., Li, Z., Wong, R.C.W., 2012. An efficient method for privacy preserving location queries. Frontiers of Computer Science 6, 409–420.

[28] Liu, Z., Li, J., Chen, X., Li, J., Jia, C., 2013. New privacy-preserving location sharing system for mobile online social networks, in: Xhafa, F., Barolli, L., Nace, D., Venticinque, S., Bui, A. (Eds.), 3PGCIC, IEEE. pp. 214–218.
[29] Mitzenmacher, M., 2002. Compressed bloom filters. IEEE/ACM Trans. Netw. 10, 604–612.
[30] Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., Boneh, D., 2011. Location privacy via private proximity testing, in: NDSS, The Internet Society.
[31] Nielsen, J.D., Pagter, J.I., Stausholm, M.B., 2012. Location privacy via actively secure private proximity testing, in: PerCom Workshops, IEEE. pp. 381–386.
[32] Nohara, Y., Inoue, S., Yasuura, H., 2008. A secure high-speed identification scheme for rfid using bloom filters, in: ARES, IEEE Computer Society. pp. 717–722.
[33] Paillier, P., 1999. Public-key cryptosystems based on composite degree residuosity classes, in: EUROCRYPT, Springer. pp. 223–238.
[34] Palmieri, P., Calderoni, L., Maio, D., 2014. Spatial bloom filters: Enabling privacy in location-aware applications, in: Lin, D., Yung, M., Zhou, J. (Eds.), Information Security and Cryptology - 10th International Conference, Inscrypt 2014, Beijing, China, December 13-15, 2014, Revised Selected Papers, Springer.
[35] Pan, X., Meng, X., 2013. Preserving location privacy without exact locations in mobile services. Frontiers of Computer Science 7, 317–340.
[36] Saldamli, G., Chow, R., Jin, H., Knijnenburg, B.P., 2013. Private proximity testing with an untrusted server, in: WISEC, ACM. pp. 113–118.
[37] Shao, M., Zhu, S., Zhang, W., Cao, G., Yang, Y., 2009. pdcs: Security and privacy support for data-centric sensor networks. IEEE Trans. Mob. Comput. 8, 1023–1038.
[38] Shokri, R., Freudiger, J., Jadliwala, M., Hubaux, J.P., 2009. A distortion-based metric for location privacy, in: Al-Shaer, E., Paraboschi, S. (Eds.), WPES, ACM. pp. 21–30.
[39] Shokri, R., Theodorakopoulos, G., Boudec, J.Y.L., Hubaux, J.P., 2011. Quantifying location privacy, in: IEEE Symposium on Security and Privacy, IEEE Computer Society. pp. 247–262.
[40] Shu, X., Yao, D.D., 2012. Data leak detection as a service, in: Keromytis, A.D., Pietro, R.D. (Eds.), Security and Privacy in Communication Networks - 8th International ICST Conference, SecureComm 2012, Springer. pp. 222–240.
[41] Son, J.H., Luo, H., Seo, S.W., 2005. Authenticated flooding in large-scale sensor networks, in: MASS, IEEE.
[42] Sun, J., Zhang, R., Zhang, Y., 2013. Privacy-preserving spatiotemporal matching, in: INFOCOM, IEEE. pp. 800–808.
[43] Tonicelli, R., David, B.M., de Morais Alves, V., 2011. Universally composable private proximity testing, in: ProvSec, Springer. pp. 222–239.
[44] von Watzdorf, S., Michahelles, F., 2010. Accuracy of positioning data on smartphones, in: LocWeb, ACM. p. 2.
[45] Wicker, S.B., 2012. The loss of location privacy in the cellular age. Commun. ACM 55, 60–68.
[46] Zakhary, S., Radenkovic, M., Benslimane, A., 2013. The quest for location-privacy in opportunistic mobile social networks, in: IWCMC, IEEE. pp. 667–673.
[47] Zheng, Y., Li, M., Lou, W., Hou, Y.T., 2012. Sharp: Private proximity test and secure handshake with cheat-proof location tags, in: ESORICS, Springer. pp. 361–378.