Spatial Bloom Filters: Enabling Privacy in Location-aware Applications

Paolo Palmieri[1], Luca Calderoni[2,⋆], and Dario Maio[2]

[1] Parallel and Distributed Systems Group
Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
`p.palmieri@tudelft.nl`

[2] Department of Computer Science and Engineering
Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
{`luca.calderoni, dario.maio`}`@unibo.it`

Abstract. The wide availability of inexpensive positioning systems made it possible to embed them into smartphones and other personal devices. This marked the beginning of location-aware applications, where users request personalized services based on their geographic position. The location of a user is, however, highly sensitive information: the user's privacy can be preserved if only the minimum amount of information needed to provide the service is disclosed at any time. While some applications, such as navigation systems, are based on the users' movements and therefore require constant tracking, others only require knowledge of the user's position in relation to a set of points or areas of interest. In this paper we focus on the latter kind of services, where location information is essentially used to determine membership in one or more geographic sets. We address this problem using Bloom Filters (BF), a compact data structure for representing sets. In particular, we present an extension of the original Bloom filter idea: the Spatial Bloom Filter (SBF). SBF's are designed to manage spatial and geographical information in a space efficient way, and are well-suited for enabling privacy in location-aware applications. We show this by providing two multi-party protocols for privacy-preserving computation of location information, based on the known homomorphic properties of public key encryption schemes. 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.

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

⋆ Part of this research work was accomplished while visiting the Parallel and Distributed Systems group of Delft University of Technology (The Netherlands).

1 Introduction

In recent years, location-aware applications spread widely. These 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 [5]. In general, the ability to detect movements and exact position of the users in outdoor environments has become widespread since the introduction of smartphones equipped with positioning systems such as a GPS receiver.

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]. 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 [11]. 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 [21], where a marketing company database model is used in conjunction with anonymous mobile phone location traces. Smartphones and location-aware services are now an integral part of our everyday life, but it is reasonable to predict that in the coming years users will demand privacy safeguards for their information with respect to the involved service provider [17] and more specifically for location information [10, 22]. The real challenge is therefore how to protect user's privacy without losing the ability to deliver services based on her location [15].

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 [5]. If we add to this scenario the requirement of the user position to stay private, the problem becomes an interesting and fundamental research question. In literature a similar problem, known as *private proximity testing* has been studied: Alice can test if she is close to Bob without either party revealing any other information about their location [12]. 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 [12]. His protocol was later improved in efficiency by Saldamli et al. [16]. Location tags and proximity tests are also used in [8], as a way of providing local authentication, while [23] 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 [13]. In [19], 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 [18].

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 [9].

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.

The novel variant of Bloom filters we introduce, which we call *Spatial Bloom Filter* (SBF), is specifically designed to deal with location 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 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. In both settings we do not assume any trust between the different parties involved. 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, unlike other works on location privacy, which is usually discussed from the end-user point of view, we work here in the secure multi-party computation, where both parties have an interest in keeping their information private. Military and government applications are just the most immediate examples of when location privacy represents a key-problem for the provider as well.

In the paper, we discuss the security and the computational cost of the proposed schemes, as well the probabilistic and storage properties of the SBF.

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. 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* $S = \{a_1, \ldots, a_n\}$ *as the set*

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

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 random 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

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

A detailed discussion about Bloom filters construction and verification as reported in literature, along with an explanation of false positive probability and optimal number of hash functions is proposed in Appendix A.

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 $(p_1 + p_2)$:

$$\mathrm{Dec}(\mathrm{Enc}(p_1) \boxplus \mathrm{Enc}(p_2)) = p_1 + p_2 \quad . \tag{3}$$

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:

$$\mathrm{Dec}(\mathrm{Enc}(p_1) \boxdot p_2) = p_1 \cdot p_2 \quad . \tag{4}$$

An example of encryption scheme that is both additively and multiplicatively homomorphic is the Paillier cryptosystem [14]. 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 entrywise) 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: Private Hadamard product of an encrypted binary vector for a cleartext vector of natural numbers

```
Input Alice: $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, $\mathbf{X} \in \{0,1\}^n$.
Input Bob: $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$, $\mathbf{Y} \in \mathbb{N}^n$.
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 $\mathbf{E} = (\mathrm{Enc}(\mathbf{x}_1), \dots, \mathrm{Enc}(\mathbf{x}_n))$.
3 Bob computes the vector $\mathbf{C} = (\mathrm{Enc}(\mathbf{x}_1) \boxdot \mathbf{y}_1, \dots, \mathrm{Enc}(\mathbf{x}_n) \boxdot \mathbf{y}_n)$ and sends
  the result to Alice.
4 Alice uses her secret key to decrypt $\mathbf{C}$ and obtains $\mathbf{D} = \mathrm{Dec}(\mathbf{C}) = \mathbf{X} \cdot \mathbf{Y}$.
```

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.

Security Model We assume the parties are *honest-but-curious*, 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 areas, and we identify a position as

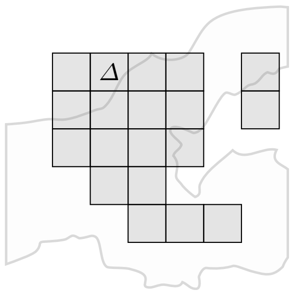


Fig. 1. A sample area covered by an arbitrary grid.

the corresponding element in this set. Considering that we can set the dimension of such areas 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. A detailed discussion concerning spatial model designed for the presented method is provided in Appendix B.

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. Appendix B 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 may, for instance, represent a sensitive or interesting location for the purposes of the application. A number of concentric areas around a point of interest (PoI) can be used to detect the user's vicinity to the PoI. In the following we present two ways in which the set-based location information described above can be used to achieve this goal. Both approaches are used in the following of the paper as strategies to select the areas of interest

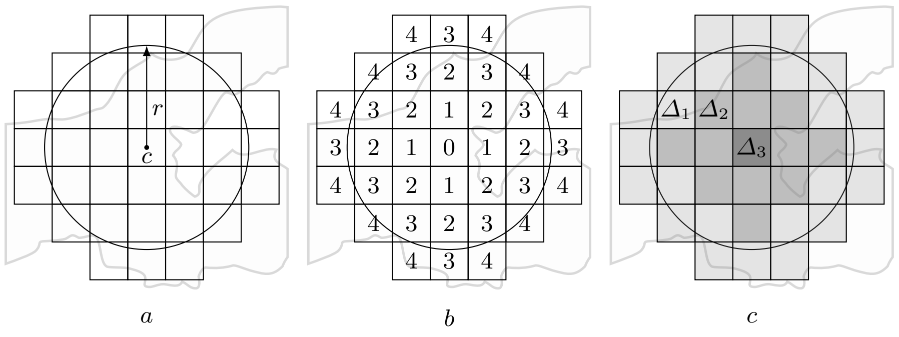


Fig. 2. An example of the area coverage algorithm applied to a point of interest of interest. After defining the grid (a), the Manhattan distance from the center region is computed (b). Finally, each region is assigned to the right set (c). In this case, the maximum distance (σ) is 4, so we assign the 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.

by the service provider, but we stress here that our construction is independent of the strategy used, and therefore can accommodate any other set selection mechanism.

The first strategy to define a set of areas of interest follows naturally from the idea of detecting the presence/absence of a person in a given area. In order to do that, the provider of the service defines an area by selecting a subset of $\mathcal{E}$ (Figure 1). The regions in the area need not to be contiguous, and there is no limitations in shape or size of the area. The set containing all of these regions is defined as Δ (we get back on this subject in Section 4).

A second approach is instead to monitor the user by detecting his proximity to the area's center as he approaches it. We achieve this goal without knowing the user's exact location by defining several concentric areas around the point of interest to be monitored. In the example shown in Figure 2 we use three areas for this purpose, but this parameter can take any value deemed useful.

Let c be the center of the area (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*. 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. 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.

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).

An important property of SBF is that 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 random number uniformly chosen in $\{1, \ldots, m\}$. We define the* Spatial Bloom Filter *(SBF) over (S, O) as the set of pairs*

$$B^{\#}(S, O) = \bigcup_{i \in I} \langle i, \max L_i \rangle \quad , \tag{5}$$

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

$$I = \bigcup_{\delta \in \bar{S}, h \in H} h(\delta) \quad , \tag{6}$$

and L_i is the set of labels l such that:

$$L_i = \{l \mid \exists \delta \in \Delta_l, \exists h \in H : h(\delta) = i\} \quad . \tag{7}$$

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

$$b^{\#}[i] = \begin{cases} l & \text{if } \langle i, l \rangle \in B^{\#}(S, O) \\ 0 & \text{if } \langle i, l \rangle \notin B^{\#}(S, O) \end{cases} \quad . \tag{8}$$

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 2 and depicted in Figure 3.

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

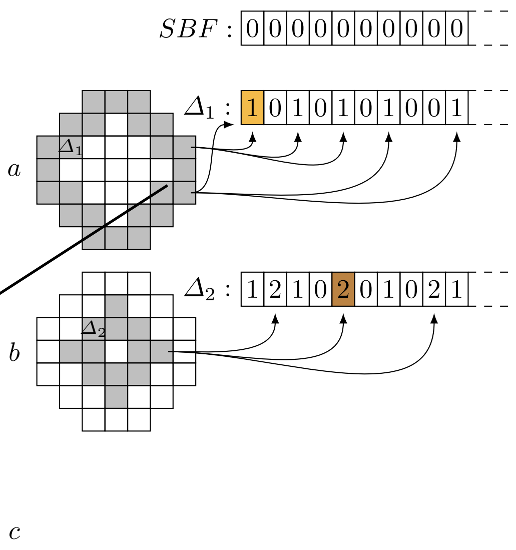
$$\exists h \in H : b^{\#}[h(\delta_u)] = i \quad \text{and} \quad \forall h \in H, b^{\#}[h(\delta_u)] \geq i \quad . \tag{9}$$

The procedure is described in Algorithm 3.

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 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 $\bar{S}$

Fig. 3. 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.



SBF : 0 0 0 0 0 0 0 0 0 0
Δ_1 : 1 0 1 0 1 0 1 0 0 1
a
Δ_1
Δ_2 : 1 2 1 0 2 0 1 0 2 1
b
Δ_2
c

Algorithm 2: Spatial Bloom Filter construction.

```
Input: $\Delta_1$, $\Delta_2$, ..., $\Delta_s$, $H$;
Output: $b^{\#}$;
1 for $i \leftarrow 1$ to $s$ do
2     foreach $\delta \in \Delta_i$ do
3         foreach $h \in H$ do
4             $b^{\#}[h(\delta)] \leftarrow i$;
          end
      end
  end
5 return $b^{\#}$;
```

Algorithm 3: Spatial Bloom Filter verification.

```
Input: $b^{\#}$, $H$, $\delta_u$, $s$;
Output: $\Delta_i$;
1 $i = s$;
2 foreach $h \in H$ do
3     if $b^{\#}[h(\delta_u)] = 0$ then
4         return $false$;
      else
5         if $b^{\#}[h(\delta_u)] < i$ then
6             $i \leftarrow b^{\#}[h(\delta_u)]$;
          end
      end
  end
7 return $\Delta_i$;
```

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 [7, 6], 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 for Spatial Bloom Filters

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 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.

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. 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

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

```
Before any communication, the provider selects the areas of interest
$\varDelta_1, \dots, \varDelta_s \subset \mathcal{E}$. Then, he selects the desired false positive probability $p$, and
determines $k$ and $m$ according to (16) and (17) respectively. Finally,
following the notation of Definition 3, the provider computes the spatial
Bloom filter $b^{\#}$ over $\bar{S}$ using Algorithm 2.
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 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 2 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 $\varDelta_i$ identifying Ursula's
  position (minus error probability $p_i$), is the smallest non-zero value in
  Dec $\left(e^{\#}\right)$.
```

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 $i \in \{1, \dots, s\}$, 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 (12) 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 $e^{\#} = \mathrm{Enc}\left(b^{\#}\right) \boxdot b_u^{\#}$ 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 $b_u^{\#}$ [9]. 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-anonimity 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 $w \in \{1, \ldots, z\}$ 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

$$\bar{a} = \frac{|\mathcal{E}|}{\sum_{w=1}^{k} \binom{s+w-1}{w} + 1} \ , \tag{13}$$

if we assume a linear distribution of the values $\{1, \ldots, s\}$ 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 and Appendix B).

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.

Computation and Communication Analysis The computational complexity for the insertion and the verification of a single element in a SBF are 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.

Since we intend this primitive to be used in concrete scenarios, in the following we provide an evaluation of actual communication costs, and number of computational operations to be performed during the execution of the protocol (Table 1). While being a generally compact data structure, a SBF built over a significantly large number of sets can consume a sizeable amount of memory. While m bits are required for storing a classical Bloom filter b, a SBF needs more bits due to the labels relative to the subsets Δ_i. 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. For instance, 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 approximately 24 KB data structure. For this reason, we introduce in the next section a protocol involving a third party which offloads user's bandwidth consumption.

5.2 Three-party Scenario

In the three-party scenario the communication does not happen directly between the service provider and the user. 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

Table 1. Computation and communication load for stakeholders.

	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)	$\mathcal{O}(m)$ $(\lfloor \log_2 s \rfloor + 1)\, m$	$\mathcal{O}(m)$ $(\lfloor \log_2 s \rfloor + 1)\, m$	
Comm. (3-p)	$\mathcal{O}(\log_2 m)$ $k\,(\lfloor \log_2 m \rfloor + 1)$	$\mathcal{O}(m)$ $(\lfloor \log_2 s \rfloor + 1)\, m$	$\mathcal{O}(m)$ $k\,(\lfloor \log_2 m \rfloor + 1) + (\lfloor \log_2 s \rfloor + 1)\, m$

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.

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
  $\mathrm{Enc}\left(b^{\#}\right)$ 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 $\{v_1, \ldots, v_k\}$ where
  $v_i = h_i\left(e_u\right)$, and sends them to Olga.
4 Olga receives the values from Ursula and builds $b_o^{\#}$, by assigning $b_o^{\#}\left[v_i\right] = 1$
  for every $v_i \in \{v_1, \ldots, v_k\}$. Then, she calculates $z$ as the number of 1's in $b_o^{\#}$.
5 Olga computes $e^{\#} = \mathrm{Enc}\left(b^{\#}\right) \boxdot b_o^{\#}$ 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 $\Delta_i$ (minus error
  probability $p_i$), is the smallest non-zero value in $\mathrm{Dec}\left(e^{\#}\right)$.
```

6 Conclusions

In this paper we present a novel privacy-preserving primitive, the spatial Bloom filter. 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 represented in a set-based format. We provide a spatial representation system for geographic areas, which allows us to encode or query positioning information (such as the one produced by GPS devices) into an SBF. In this paper we propose two protocols for privacy preservation in location-based services based on spatial Bloom filters. We imagine a scenario in which the provider of the service is interested in detecting the presence of a user within predetermined areas of interest, or his proximity to points of interest. Thanks to the properties of SBF, the provider can build a filter over a number of geographic areas, and the user can query the filter to determine whether his current location lies within those areas without knowing the areas themselves, thus preserving the privacy of the provider. By using the homomorphic properties of a public key encryption scheme, we can also guarantee the user's privacy, by allowing the provider to only learn in which (predefined) area the user is, and not his exact position. The provider learns nothing in case the user is outside the predefined set of areas.

Acknowledgments

The authors would like to acknowledge Marco Miani for the code used in producing Figure 4.

References

1. Avoine, G., Calderoni, L., Delvaux, J., Maio, D., Palmieri, P.: Passengers information in public transport and privacy: Can anonymous tickets prevent tracking? Int J. Information Management 34(5), 682–688 (2014)
2. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
3. Blum, J.R., Greencorn, D.G., Cooperstock, J.R.: Smartphone sensor reliability for augmented reality applications. In: MobiQuitous. LNICST, vol. 120, pp. 127–138. Springer (2012)
4. Blumberg, A.J., Eckersly, P.: On Locational Privacy, and How to Avoid Losing it Forever. https://www.eff.org/wp/locational-privacy (April 2009)
5. Calderoni, L., Maio, D., Palmieri, P.: Location-aware mobile services for a smart city: Design, implementation and deployment. JTAER 7(3), 74–87 (2012)
6. Charles, D.X., Chellapilla, K.: Bloomier filters: A second look. In: Halperin, D., Mehlhorn, K. (eds.) ESA. Lecture Notes in Computer Science, vol. 5193, pp. 259–270. Springer (2008)
7. Chazelle, B., Kilian, J., Rubinfeld, R., Tal, A.: The bloomier filter: an efficient data structure for static support lookup tables. In: SODA. pp. 30–39. SIAM (2004)
8. Jiazhu, D., Zhilong, L.: A location authentication scheme based on proximity test of location tags. In: ICINS 2013. pp. 1–6 (Nov 2013)
9. Kikuchi, H., Sakuma, J.: Bloom filter bootstrap: Privacy-preserving estimation of the size of an intersection. JIP 22(2), 388–400 (2014)
10. Kulik, L.: Privacy for real-time location-based services. SIGSPATIAL Special 1(2), 9–14 (2009)
11. de Montjoye, Y.A., Hidalgo, C.A., Verleysen, M., Blondel, V.D.: Unique in the Crowd: The privacy bounds of human mobility. Scientific Reports 3 (March 2013)
12. Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., Boneh, D.: Location privacy via private proximity testing. In: NDSS. The Internet Society (2011)
13. Nielsen, J.D., Pagter, J.I., Stausholm, M.B.: Location privacy via actively secure private proximity testing. In: PerCom Workshops. pp. 381–386. IEEE (2012)
14. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. LNCS, vol. 1592, pp. 223–238. Springer (1999)
15. Pan, X., Meng, X.: Preserving location privacy without exact locations in mobile services. Frontiers of Computer Science 7(3), 317–340 (2013)
16. Saldamli, G., Chow, R., Jin, H., Knijnenburg, B.P.: Private proximity testing with an untrusted server. In: WISEC. pp. 113–118. ACM (2013)
17. Shu, X., Yao, D.D.: 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. LNCS, vol. 106, pp. 222–240. Springer (2012)
18. Sun, J., Zhang, R., Zhang, Y.: Privacy-preserving spatiotemporal matching. In: INFOCOM. pp. 800–808. IEEE (2013)
19. Tonicelli, R., David, B.M., de Morais Alves, V.: Universally composable private proximity testing. In: ProvSec. LNCS, vol. 6980, pp. 222–239. Springer (2011)

20. von Watzdorf, S., Michahelles, F.: Accuracy of positioning data on smartphones. In: LocWeb. p. 2. ACM (2010)
21. Wicker, S.B.: The loss of location privacy in the cellular age. Commun. ACM 55(8), 60–68 (2012)
22. Zakhary, S., Radenkovic, M., Benslimane, A.: The quest for location-privacy in opportunistic mobile social networks. In: IWCMC. pp. 667–673. IEEE (2013)
23. Zheng, Y., Li, M., Lou, W., Hou, Y.T.: Sharp: Private proximity test and secure handshake with cheat-proof location tags. In: ESORICS. LNCS, vol. 7459, pp. 361–378. Springer (2012)

A Bloom Filters Properties

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

$$\forall h \in H, b\left[h(a_u)\right] = 1 \ . \tag{14}$$

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:

$$p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \approx \left(1 - e^{-\frac{kn}{m}}\right)^{k} \ . \tag{15}$$

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 (15), we can calculate the optimal number of hashes k as

$$\operatorname{opt}(k) = \frac{m}{n} \ln 2 \ , \tag{16}$$

from which we can infer

$$m = \left\lceil -\frac{n \ln p}{(\ln 2)^2} \right\rceil \ . \tag{17}$$

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.

B More on Spatial Representation

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 known as the *graticule* (Figure 4).

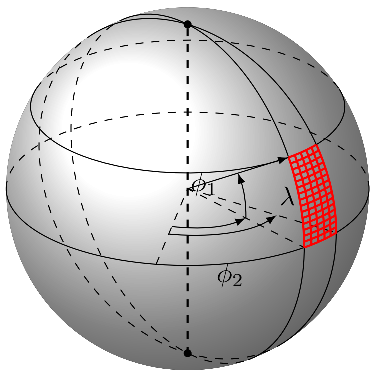


Fig. 4. An example of the planet's surface and the grid plotted on it. ϕ_1 and ϕ_2 are longitude values while λ is a latitude value.

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 4, the portions of the Earth's surface defined by such a grid have varying areas depending on their position (Table 2). While the construction proposed in the following is not dependent on the size or shape of the regions, for simplicity in the discussion it is reasonable to approximate such portions to rectangles and assume they have the same area.

In actual applications, the precision in decimal points 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.) [20].

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

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 2. Some reference values of accuracy using three decimal places for coordinate representation.

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.