Map-aware Position Sharing for Location Privacy in Non-trusted Systems

Pavel Skvortsov, Frank Dürr, Kurt Rothermel
{pavel.skvorzov,frank.duerr,kurt.rothermel}@ipvs.uni-stuttgart.de

Institute of Parallel and Distributed Systems, Universität Stuttgart

Abstract. Many current location-based applications (LBA) such as friend finder services use information about the positions of mobile users. So-called location services (LSs) have been proposed to manage these mobile user positions efficiently. However, managing user positions raises privacy issues, in particular, if the providers of LSs are only partially trusted. Therefore, we presented the concept of private position sharing for partially trusted systems in a previous paper [1]. The basic idea of position sharing is to split the precise user position into a set of position shares of well-defined limited precision and distribute these shares among LSs of different providers.
The main contributions of this paper are two extended position sharing approaches that improve our previous approach in two ways: Firstly, we reduce the predictability of share generation that allows an attacker to gain further information from a sub-set of shares to further increase the position precision. Secondly, we present a position sharing algorithm for constrained movement scenarios whereas the existing approach was tailored to open space environments. However, open space approaches are vulnerable to map-based attacks. Therefore, we present a share generation algorithm that takes map knowledge into account.

Keywords: Location-based service; privacy; obfuscation; sharing; map-awareness.

1 Introduction

Location services (LS) such as Yahoo! Fire Eagle, Google Latitude, InstaMapper, or Trace4Youstoring the positions of mobile users have become an important prerequisite for many advanced location-based applications (LBA). In particular, LS are beneficial if various LBAs have to be provided with the position of the mobile user. For instance, the position of a user could be accessed by several social networks like Facebook and Gowalla, a friend alert service, a location-based advertising service, a traffic congestion service, etc. Obviously, in such scenarios the LS, which usually runs on powerful servers, relieves the mobile device from sending individual positions to each LBA.

However, since user positions are privacy-sensitive information, problems might arise if the LS provider is not fully trusted by the mobile user. This might be the case for various reasons. For instance, the provider might be malicious and misuse data, e.g., selling it to another party. However, even if the provider itself is not malicious, he might simply be unable to protect the data from attacks. As various examples in the past show, even infrastructures that were deemed to be trustworthy were subject to successful attacks, information leaks, loss of data, etc. [2], [3], [4]. With the advent of cloud computing, we are likely to see even more cloud-based location services in the future operated by non-trusted providers. Using these services is certainly attractive from a monetary and technical point of view, however, it requires concepts to reduce privacy risks.

Besides LSs, LBAs might also be only partially trusted by the mobile user. However, LBAs have to be provided with position information of a certain quality to provide the requested functionality. This leads to a situation, where the user might want to trade-off his privacy in terms of the precision of positions provided to LBAs and the quality of service provided by the LBA. For instance, it might be necessary to provide a friend alert service with positions of a precision of 200 m, whereas 1 km precision might be sufficient for Facebook to show location-based status messages. In other words, the user might want to define different *privacy levels* corresponding to different degrees of precision individually for each LBA.

In [1], we proposed the concept of *position sharing* to solve the above privacy issues. The basic idea is that the (trusted) mobile device of the tracked user splits up the precise position information into so-called position shares of limited precision. These shares are distributed among a set of LSs of *different* providers, i.e., each LS only manages position information of strictly limited, well-defined precision. The mobile users provides LBAs with access rights to access a certain number of position shares from different LSs. Using a so-called share fusion algorithm, these shares can be combined to yield a position of a certain precision which is higher than the precision of the single shares stored at the LSs. Fusing all shares restores the precise position. Therefore, different privacy levels can be defined for individual applications by providing the LBA with access rights to a certain number of shares. Besides the possibility to define different privacy levels, this approach has another important advantage: it has a graceful degradation of privacy property since a compromised LS will only reveal information of strictly limited precision. Therefore, it is possible to utilize not fully trusted LSs to store position information and still limit the risk since no single point of failure exists w.r.t. privacy.

In this paper, we are going to improve our previous system further. As a first main contribution, we propose a new share generation algorithm that increases location privacy by reducing the predictability of share generation. Therefore, it becomes harder for an attacker to derive more precise positions than intended from a certain number of shares. Secondly, our first approach only targeted free space environments where users can move without restriction. However, in highly structured areas such as cities this makes it easy for an attacker to restrict and therefore predict positions of high precision. Therefore, as the second main contribution of this paper, we propose a novel *map-aware position sharing approach* that considers movement restrictions given by the physical environment. The basic idea is to adapt the size of obfuscation areas defined by shares based on map information. Therefore, a sufficiently large area of possible positions is retained also under movement restrictions.

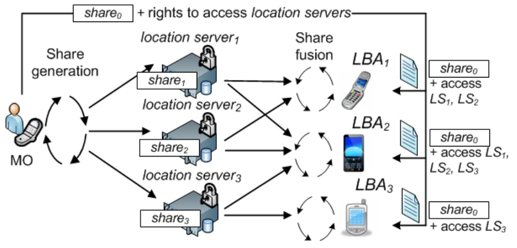
The rest of this paper is structured as follows. First, in Section 2 we present our system model, problem statement and formal definition of obfuscation security. In Section

3 we describe the basic position sharing approach and two extended approaches including the new map-aware position sharing. Then in Section 4 we analyze the security of obfuscation provided by the our approach. After that we overview the related work in Section 5. Finally, we conclude the results of our work and give some perspective for the future research.

2 System Model and Problem Statement

In this section, we introduce the different components of our location management system together with the formal notation used in this paper.

Our system consists of three types of components: mobile objects, location servers, and location-based applications (see Figure 1).



$share_0$ + rights to access *location servers*
Share generation
MO
location server$_1$
$share_1$
location server$_2$
$share_2$
location server$_3$
$share_3$
Share fusion
LBA_1
LBA_2
LBA_3
$share_0$ + access LS_1, LS_2
$share_0$ + access LS_1, LS_2, LS_3
$share_0$ + access LS_3

Fig. 1. System model

Mobile objects (MO) are the objects whose positions are managed on location servers and used by location-based applications. MOs correspond to users carrying a mobile device such as a smart phone with a position system like GPS. Using this positioning system, the MO can determine its current position, denoted as p_{user}. For the sake of simplicity, we assume p_{user} to be perfectly precise and accurate. This is a reasonable simplification if we assume that the imprecision introduced by position obfuscation is much greater than the imprecision of the positioning system. However, the algorithms could easily be extended to also consider the intrinsic imprecision of the positioning system.

On the MO, a trustworthy component is located that runs a share generation algorithm. Given a precise position p_{user}, a number of n shares, and a lowest precision ϕ_{min}, the share generation algorithm generates a set of position shares denoted as $S = (s_0, \{s_1, \dots, s_n\})$:

$$\text{generate}(p_{user}, n, \phi_{min}) = S \tag{1}$$

In more detail, S consists of a so-called master share s_0 which represents a position with lowest precision ϕ_{min}. $s_1, \dots, s_n$ are called refinement shares. Given s_0 and a set $S_k \subseteq S$ of k refinement shares, a refined position p_k can be calculated using a share fusion algorithm:

$$\text{fuse}(s_0, S_k) = p_k \tag{2}$$

The basic idea is that each refinement share increases the precision by a well-defined value Δ_ϕ. That is, fusing k refinement shares and the master share yields a position p_k of higher precision $\phi_k = \phi_{min} - k\Delta_\phi$, where $\phi_k < \phi_{k-1}$. Fusing all shares reconstructs the precise position p_{user}.

The precision ϕ is represented as a radius, if the obfuscation area is circular. However, later we will see that obfuscation areas do not need to be circular areas. Instead, an obfuscation area can have any shape. Therefore, we additionally measure the precision as the size of the area[1]. Thus, we say that a higher precision corresponds to a smaller radius and a smaller area, while a lower precision corresponds to larger radius and larger area of the obfuscation shape.

After share generation, the MO distributes the shares among a set of location servers (LS), denoted as L, where every LS is operated by a different provider. Formally, share distribution is defined as an injective function that distributes every refinement share to an unique LS:

$$\text{distribute}(\{s_1, \ldots, s_n\}, L) : S \rightarrow L \tag{3}$$

The master share is known to everybody — in particular, every location-based application, — for instance, through full replication at every LS and unrestricted access by location-based applications. Hence, every location-based application can track MOs with (at least) a precision of ϕ_{min}. Therefore, ϕ_{min} is usually chosen large, i.e., with a low precision.

Refinement shares are only known to authorized location-based applications (LBA). The MO specifies, which precision each LBA should get. Usually, this decision defines a tradeoff between the quality of service an LBA can provide with a certain precision of information, and the privacy requirements of the MO. The trusted share generation component running on the MO's device determines, how many refinement shares from the set of refinement shares are required for this precision using the above formula for ϕ_k. Then, the MO assigns access rights to refinement shares. Shares and the respective access rights are sent together to the LSs. LSs use common access control mechanisms to deliver refinement shares only to authorized LBAs. LBAs receive the necessary access rights (credentials) together with the relevant LS addresses from the MO.

The injective mapping ensures that each LS only knows exactly one refinement share (plus the master share). Therefore, a compromised LS reveals only one refinement share equivalent to a position of strictly limited precision. The precision available to an attacker increases linear with the number of compromised servers. This ensures the most important property of the approach: graceful degradation of privacy (increase of precision) with the number of compromised shares.

Theoretically, the mapping of shares to LS or the precision increase Δ_ϕ could be adjusted to the individual trustworthiness of the LS, giving more trusted LSs better or more shares. The individual trust value could be defined by a trust management system, for which several concepts have been described in the literature [5]. However, such optimizations are beyond the scope of this paper. Here, we simply assume that every server stores one refinement share per position, and every refinement share has the same precision increase Δ_ϕ.

[1] In order to make both precision metrics comparable, it is possible to convert the area of a non-circular obfuscation shape to a radius of a virtual circle with the size of the area.

We also assume that each MO stores a map of the environment. For each location, a map contains a binary Boolean value: true if the MO could be located at this location (e.g., a street, building, etc.); false otherwise (e.g., a lake, agriculture field, etc.). As shown later, this map knowledge is used by the share generation algorithm to generate shares that do not allow for the derivation of position information of higher precision than intended. A simple example of a binary map representation derived from a given map (Figure 2a) is presented in Figure 2b.

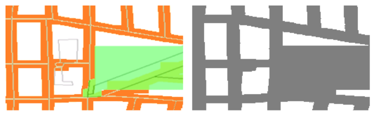


Fig. 2. (a) Basic map: roads and squares; (b) Binary representation of the map knowledge

Finally, we assume that share generation is only triggered sporadically rather than with every update of the positioning system. Typically, this is the case when using a "check-in" usage pattern, where the user manually publishes her position sporadically at certain locations. Although the presented algorithms could also work with continuous positions updates, subsequent (close) positions might reveal additional information to an attacker. Such problems arising from continuous updates are beyond the scope of this paper. Instead we assume that a minimum position update interval is ensured that guarantees that succeeding obfuscation shapes of precision ϕ_k do not intersect.

Problem Statement and Obfuscation Security Metrics. The problem is to find a secure share generation algorithm (implementation of function $\mathrm{generate}(p_{user}, n, \phi_{min})$) such that the following property is fulfilled for the generated shares: Given the master share s_0 and a set S_k of refinement shares, it should not be possible to derive a position with higher precision than the intended precision $\phi_k = \phi_{min} - k\Delta_\phi$. Informally, we say that the generated shares are insecure. Obviously, if this would be possible, the MO could not effectively control the precision of information offered to LBAs by assigning access rights to a certain number (k) of shares corresponding to the intended precision (ϕ_k).

The above statement is very strict considering the fact that an implementation of a share generation algorithm might always lead to a certain degree of insecurity. The best we can do is to design an algorithm that *minimizes* the insecurity of shares as much as possible. Therefore, we introduce metrics, called *obfuscation security metrics*, that quantify the degree of share security. As we will see later, we use probabilistic share generation algorithms. Therefore, our security metrics are also probabilistic metrics expressing the probability that an attacker can reveal a position of a certain precision. We propose two metrics:

Let s_0 and S_k be a master share and a set of refinement shares, and p_k be the area resulting from the fusion of these k shares. Then the obfuscation security metrics $P(\phi_{k,\mathrm{attack}})$ defines the probability that an attacker can refine the MO's position to an area with precision $\phi_{k,\text{attack}}$ where $\phi_{k,\text{attack}} \leq \phi_k$. A perfectly secure set of shares would lead to a probability of 0.0 for every $\phi_{k,\text{attack}} \leq \phi_k$. This metrics gives insight into the absolute precision that an attacker can gain, for instance, an attacker can calculate a position of 500 m precision with 90% probability.

The second obfuscation security metric, $P_{10\%}$, defines the probability that an attacker can pinpoint the MO's position to an area of 10% size of p_k. A perfectly secure set of shares leads to $P_{10\%} = 0.1$. Or in other words: The position of MO is uniformly distributed within the obfuscation area. A non-uniform probability distribution of MO within p_k increases $P_{10\%}$ to values greater than 0.1. This metrics is based on a relative area size compared to p_k. It has to be noted that the choice of using 10% instead of another value is somewhat arbitrary reflecting the case that the attacker can gain a position of much higher precision than intended. We could also use any other relative area size.

3 Position Sharing Approaches

In this section, we present three different position sharing approaches. As basis, we start with a brief description of our basic approach published in [1], called OSPS-ASO (Open Space Position Sharing with Any Share Order). Then, we improved this approach in two ways: Firstly, the share generation of OSPS-ASO is (to a certain degree) predictable, i.e., from k out of n shares information with a higher precision than the intended precision $\phi_{min} - k\Delta_\phi$ can be derived. Therefore, we present an improved share generation algorithm called OSPS-FSO (Open Space Position Sharing with Fixed Share Order), which decreases the predictability of shares. Secondly, OSPS-ASO does not consider map knowledge, which can also be used by an attacker to increase the precision unintentionally. To solve this problem, we present a map-aware share generation algorithm called CSPS (Constrained Space Position Sharing), which considers movement restrictions given by the physical environment.

3.1 Basic Approach: OSPS-ASO

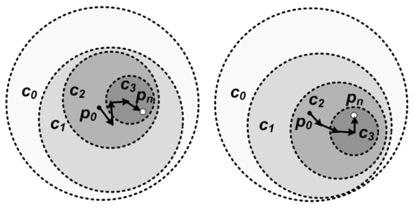


Fig. 3. OSPS-ASO: fusion of the same set of shares in different order

In order to explain how OSPS-ASO implements the concept of position sharing, we first give a detailed definition of shares and explain the share fusion algorithm, before we explain the generation of shares.

Algorithm 1 shows the share fusion algorithm of OSPS-ASO. With this open space approach, positions are defined as circles with radius (precision) ϕ. The master share s_0 is simply a circle c_0 with radius $r_0 = \phi_{min}$ centered at p_0. Each refinement share defines a shift vector that shifts the center of the previous obfuscation circle. At the same time, the radius is decreased by value Δ_ϕ. The fusion algorithm incrementally shifts the positions of obfuscation circles starting from p_0, and after each shift decreases the radius by Δ_ϕ.

Algorithm 1 OSPS-ASO: fusion of shares

```
1: function $fuse\_k\_shares\_OSPS\_ASO(n, c_0, \boldsymbol{s_1} \ldots \boldsymbol{s_k})$
2: $\Delta r \leftarrow r_0/n$
3: $\boldsymbol{p} \leftarrow p_0$
4: $r \leftarrow r_0$
5: for $i = 1$ to $k$ do
6:     $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{s_i}$;
7:     $r \leftarrow r - \Delta r$
8: return $c_k = \{p, r\}$
```

Note that with this approach, shares can be added in any order since the shift operation is a commutative operation (therefore the name "Any Share Order"). As a consequence, the maximal length of shift vectors has to be limited by $\Delta r = r_0/n = \Delta_\phi$. As a result, any refined circle is completely contained in the previous circle. Two examples of share fusion in different orders for $n = 4$ and $k = 3$ given refinement shares are shown in Figure 3.

The algorithm for share generation is shown in Algorithm 2. Input parameters to this algorithm are the precise user position p_{user}, radius $r_0 = \phi_{min}$ of the master share, and the total number of refinement shares n.

Algorithm 2 OSPS-ASO: generation of shares

```
1: function $generate\_n\_shares\_OSPS\_ASO(p_{user}, n, \phi_{min}, \Delta_\phi)$
2: select randomly $p_0$ such that $distance(p_0, p_{user}) \leq \phi_{min}$
3: do
4:     for $i = 1$ to $n - 1$ do
5:         select rand. $\boldsymbol{s_i}$ with $|\boldsymbol{s_i}| \leq \Delta_\phi$ such that $p_{user} \in c_i$
6: while $distance(\boldsymbol{p_0} + \sum_{i=1}^{n-1} \boldsymbol{s_i}, \boldsymbol{p_{user}}) > \Delta_\phi$
7: $\boldsymbol{s_n} \leftarrow \boldsymbol{p_{user}} - (\boldsymbol{p_0} + \sum_{i=1}^{n-1} \boldsymbol{s_i})$
8: return $\boldsymbol{s_0} \ldots \boldsymbol{s_n}$
```

In the first step, position p_0 of c_0 (master share) is selected such that p_{user} is distributed uniform at random within c_0 (2). Then we calculate $n - 1$ refinement shares. For each of these shares, we choose a random direction and random length within the valid length interval $[0, \Delta_\phi]$ (3-6). The last (nth) refinement share has to connect the end point of the concatenated $n - 1$ refinement shares to p_{user} (7). If the resulting length of the nth vector is smaller than or equal to the length constraint Δ_ϕ, we have found a valid set of refinement shares. Otherwise, we repeat the process, i.e., re-calculate $n-1$ random refinement shares, and try to connect their end point to p_{user} within the given length constraint.

3.2 Extended Approach 1: OSPS-FSO

Next, we are going to present the improved position sharing approach OSPS-FSO that improves the security of calculated shares.

The design goal of OSPS-ASO was to be able to combine shares in any order. Therefore, also a missing share does not prevent share fusion altogether. Also if refinement shares are missing, the remaining shares can still be fused to a position of increased precision where the precision increase linearly depends on the number of missing shares. This increases the robustness of OSPS-ASO w.r.t. failed LSs.

However, as we have shown in [1], the predictability of further shares can be also increased as an attacker knows more and more refinement shares. The obtained precision increases beyond the intended precision, e.g., if the constrained share generation parameters make the resulting vectors correlated. This is due to the length restriction of shift vectors, which ensures that refined obfuscation circles are completely contained within the unrefined circles.

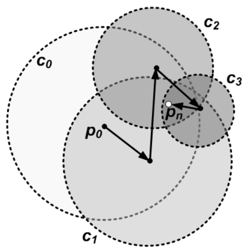


Fig. 4. Fusion of shares in a fixed order without area adjustment

We will show that the security of obfuscation can be substantially increased by combining shares only in fixed order (therefore the name "Fixed Share Order"). Unfortunately, by fixing the share order, the robustness of the scheme w.r.t LS failures of attacks is decreased. However, if the availability of location information is critical, robustness can be achieved by other means such as replication of shares.

Algorithm 3 shows the adapted share fusion algorithm of OSPS-FSO. Similar to OSPS-ASO, each refinement share defines a shift of the center of the previous circle c_{i-1}. The result is again a circle c_i with a different (predefined) radius r_i. However, since the length of the shift is not restricted anymore, c_i might not be completely contained within c_{i-1} (see Figure 4). Instead, the new obfuscated position p_k is defined by the intersection of c_i and the previous obfuscation area. Consequently, p_k might not be a circle anymore but an area defined by the intersection of multiple circles. Only the

Algorithm 3 OSPS-FSO: fusion of shares

```
1: function $fuse\_k\_shares\_OSPS\_FSO(n, c_0, \boldsymbol{s_1} \dots \boldsymbol{s_k}, r_1 \dots r_k)$
2: $A_k \leftarrow c_0$
3: $\boldsymbol{p} \leftarrow p_0$
4: for $i = 1$ to $k$ do
5:     $\boldsymbol{p} \leftarrow \boldsymbol{p} + \boldsymbol{s_i}$
6:     $c_i \leftarrow \{\boldsymbol{p}, r_i\}$
7:     $A_k \leftarrow A_k \cap c_i$
8: end for
9: return $A_k$
```

master share s_0 still must be a circle of radius $r_0 = \phi_{min}$. We denote A_k to be the size of the area resulting from the intersection of k circles, i.e., the fusion of k refinement shares and the master share:

$$A_k = \text{area}(c_0 \cap c_1 \cap \ldots \cap c_k) = \pi * r_k^2 \qquad (4)$$

We say, the obfuscated position p_k has the precision ϕ_k if $A_k = \pi * (\phi_{min} - k\Delta_\phi)^2$. That is, we compare the obfuscation area to a circle covering an area of the same size.

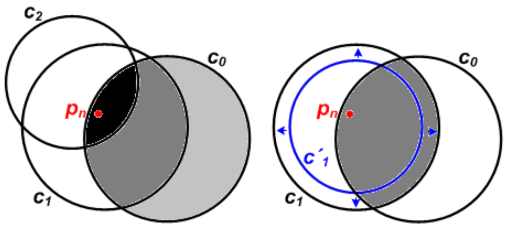


Fig. 5. OSPS-FSO: (a) intersection of 3 circles c_0, c_1, c_2; (b) adjustment of intersection area through radius increase of c_1' to c_1: $A_1 = \text{area}(c_0 \cap c_1) = \text{area}(c_1')$

Moreover, we need another modification to ensure that every additional share increases the precision by a well-defined value Δ_ϕ. We achieve this by an adjustment of the radius r_i of circle c_i such that A_i equals the area of the initial (non-adjusted) circle c_i'. Algorithm 4 shows the share generation algorithm of OSPS-FSO. In lines 8-10, we increase the radius to match the desired size A_k, as it is shown in Figure 5.

However, only increasing the radius is not sufficient for secure share generation, if an attacker knows the share generation algorithm. If we only increase radius r_i without changing the circle center p_i, an attacker can simply reduce the obfuscation area A_i by decreasing the obtained radius r_i down to the initial (non-increased) value of the radius r_i' (see Figure 6a). In order to avoid such an attack, the center of circle c_i must be adjusted so that the original position of c_i' within c_i cannot be found.

Algorithm 5 shows a secure algorithm for increasing r_i by also moving the center p_i. First, for the current p_i we determine the radius r_i which makes the intersection area

Algorithm 4 OSPS-FSO: generation of shares

```
1: function $generate\_n\_shares\_OSPS\_FSO(p_{user}, n, \phi_{min}, \Delta_\phi)$
2: select randomly $p_0$ with $distance(p_0, p_{user}) \leq \phi_{min}$
3: $A_0 \leftarrow$ area$(c_0)$
4: for $i = 1$ to $n - 1$ do
5:     $r_i \leftarrow \phi_{min} - i * \Delta_\phi$
6:     select rnd. $\boldsymbol{s_i}$ with $p_{user} \in c_i$
7:     $A_i \leftarrow$ area$(c_i)$
8:     while area$(\cap_{j=1}^{i}(c_j)) < A_i$ do
9:         $r_i \leftarrow$ increase$(r_i)$
10:    end while
11: end for
12: $\boldsymbol{s_n} \leftarrow \boldsymbol{p_{user}} - (\boldsymbol{p_0} + \sum_{i=1}^{n-1} \boldsymbol{s_i})$
13: return $\boldsymbol{s_0} \dots \boldsymbol{s_n}, r_0 \dots r_n$
```

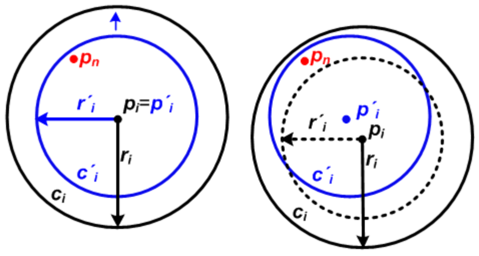


Fig. 6. Adjustment of p_i during radius increase: (a) no adjustment of p_i; (b) randomized adjustment of p_i

Algorithm 5 Radius increase with adjustment of p_i

```
1: function $increase(r_i, p_i, \Delta r, p_n)$
2: $r'_i \leftarrow r_i$
3: $A_i \leftarrow$ area$(c'_i)$
4: while area$(\cap_{j=1}^{i}(c_j)) < A_i$ do
5:     $r_i \leftarrow r_i + \Delta r$
6: end while
7: $x_{shift} \leftarrow$ get_random_shift$(p_i, r'_i, r_i)$
8: $y_{shift} \leftarrow$ get_random_shift$(p_i, r'_i, r_i)$
9: $p_i \leftarrow$ shift$(p_i, x_{shift}, y_{shift})$
10: if area$(\cap_{j=1}^{i}(c_j)) < A_i$ then
11:     $r_i \leftarrow$ increase$(r_i)$
12: else
13:     while (area$(\cap_{j=1}^{i}(c_j)) > A_i$) and $(p_n \in c_i)$ do
14:         $r_i \leftarrow r_i - \Delta r$
15:     end while
16:     $r_i \leftarrow r_i + \Delta r$
17: end if
18: return $p_i, r_i$
```

large enough; we used radius increase $\Delta r = r/20$ (4-6). Then we perform the random shift of p_i, not longer than $r_i - r'_i$ (7-9). After that we check whether the current radius r_i is satisfying the area condition (10). If the intersection area is again not large enough, we call the function **increase**$(r_i, \ldots)$ recursively (11). If the intersection area now exceeds the target value A_i, we decrease the current radius r_i until it achieves the required size (12-16).

In Figure 6b, it is shown that after the adjustment of p_i the target position p_n can be located anywhere within c_i. In other words, an attacker is not able to reduce the obfuscation area A_i just knowing the share generation algorithm.

3.3 Extended Approach 2: CSPS

Although OSPS-FSO as presented in the previous sub-section solves the problem of predictable refinement shares in an open space environment, it is not sufficient for restricted movement scenarios. If an MO cannot or is at least unlikely to move in certain areas such as lakes or agriculture fields, these areas can be subtracted from the obfuscated position p_k calculated by the share generation algorithm of OSPS-FSO. This effectively reduces the size A_k of p_k to a value below the intended precision $\phi_{min} - k\Delta_\phi$ (see Figure 7a).

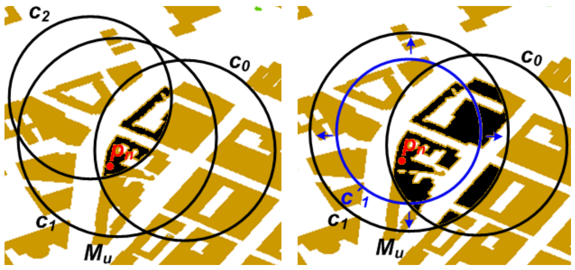


Fig. 7. CSPS: (a) intersection of 3 circles c_0, c_1, c_2 and the map representation M_u; (b) adjustment of intersection area through radius increase of c_1: $A_1 = \text{area}(M_u \cap c_0 \cap c_1) = \text{area}(c'_1)$. The black area depicts the effective obfuscation area where the user can actually be located.

Obviously, this unintended size reduction of p_k is due to the fact that OSPS-FSO does not consider such movement restrictions during share generation. This problem can be solved by considering movement restriction during radius adjustment. As prerequisite, a user-defined map M_u is required which defines possible positions of MO. Depending on the type of MO, M_u might contain general areas such as streets, shops, public places, as well as individual areas such as the user's home and working place. In general, M_u should reflect all movement restrictions that a possible attacker could know. Obviously, if an attacker is aware of additional movement restrictions that were not considered during share generation, he can possibly further reduce the obfuscation area.

Based on M_u, we can now modify the share fusion and share generation algorithm to consider movement restrictions to define our Constraint Space Position Sharing Approach (CSPS). The main idea of the map-aware share generation algorithm of CSPS is to increase in each share generation step i the radius of circle c_i, until the size[2] of $M_u \cap (c_0 \cap \ldots \cap c_i)$ is equal to $\phi_{min} - i\Delta_\phi$ (see Figure 7b). Informally, this means that we increase the intersection area until there are enough locations where the MO can *actually* be located.

Compared to the fusion algorithm of OSPS-FSO (Algorithm 3), only a small change in is required. The obfuscated position is not only defined by the intersection of the circles c_i, but also the intersection with M_u to remove areas where the user cannot be located. Thus, line 2 is modified to: $A_k \leftarrow M_u \cap c_0$.

The map-aware share generation algorithm of CSPS is similar to Algorithm 4, with additional c_0 increase before the main cycle for generating shares $s_1 ... s_n$:

```
while area($M_u \cap c_0$) < $A_0$ do
  $r_0 \leftarrow$ increase($r_0$)
end while
```

Also, the condition of line 8 of Algorithm 4 now must include M_u:

```
while area($M_u \cap \cap_{j=1}^{i}(c_j)$) < $A_i$ do
```

4 Evaluation of Obfuscation Security

In this section, we evaluate the obfuscation security provided by the share generation algorithm. For a quantitative comparison, we use the metrics introduced in Section 2. First, we introduce our attacker model before we describe the evaluation results.

Attacker Model. On the one hand, attackers include *external attackers* who try to circumvent the access control mechanisms of LSs to get access to as many secret refinement shares as possible. Since preventing such attacks basically requires well-known access control mechanisms, which are not specific to position sharing, we will not consider this kind of attack further in our evaluation.

On the other hand, attackers also include *internal attackers* in form of malicious LS or LBA providers. In general, such internal attackers have access to k out of n shares. In detail, a LS has access to 1 out of n shares, namely, the single share managed by the LS. A compromised LBA has access to the k out of n shares for which it received access rights from the MO. As already described in the beginning, k defines a trade-off between the QoS that can be offered by the LBA due to the limited precision of position information, and the degree of lost privacy should the LBA misuse the position information. Therefore, in our approach adjusting k is the basic means of controlling privacy risks. Therefore, we focus our evaluation on such internal attacks where the attacker knows k out of n shares. We should note that we do not explicitly consider

[2] The size of intersection area is calculated through space discretization: we count the number of points covered by the intersection shape and convert this number into the corresponding area value proportionally.

the case of colluding internal attacker, i.e., multiple malicious LS or LBA providers that exchange their shares to increase the number of (compromised) shares. To handle this case, the MO needs to assess the risk that providers collude, which is a different problem of defining suitable trust relations and modeling relations between providers—for instance, which LS are sharing the same server (cloud) infrastructure operated by the same third-party provider, or which providers have to reveal their data to the same legal entity because they fall under the same jurisdiction, etc.

As already mentioned, adjusting k is only then an effective means to control privacy if the precision of positions derived from these shares are well-defined. If the share generation algorithm is perfectly secure, an attacker with k compromised shares can calculate a position with at least the precision $\phi_k = \phi_{min} - k\Delta_\phi$. However, due to a certain predictability of share generation, he might even increase the precision beyond that value as already discussed in Section 3. Since we assume that the share generation algorithm is known to everybody, the attacker can use a *Monte Carlo Simulation* to simulate the process of share generation and predict further possible refinement shares from the known shares. To this end, he runs the share generation algorithm many times to sample the probability distribution of the MO position and analyzes the resulting position distribution to determine the most likely area where the MO is located in.

To quantify the (undesired) effect of share prediction and the resulting effective security of shares, we use the metrics $P_{10\%}$ and $P(\phi_{k,attack})$ already defined in Section 2.

4.1 Open Space Evaluation (OSPS-ASO vs. OSPS-FSO)

In our first evaluation, we start with the assumption that MOs can move without restrictions in an open space. This evaluation shows the difference between our old approach OSPS-ASO presented in [1] that fuses the refinement shares in arbitrary order, and the fixed order fusion approach OSPS-FSO presented in this paper.

As mentioned, in the ideal case each share should increase the precision by exactly Δ_ϕ. Depending on the predictability of share generation, the attacker can gain a higher precision $\phi_{k,attack} \leq k\Delta_\phi$ from k (compromised) shares with a certain probability $P(\phi_{k,attack})$. Figure 8 plots $\phi_{k,attack}$ over k for different probabilities $P(\phi_{k,attack})$ (the total number of refinement shares is $n = 5$; $\phi_{min} = 10000$ m). Figure 8a shows the results for our old algorithm OSPS-ASO; Figure 8b shows the results of OSPS-FSO proposed in this paper.

Obviously, with 100% probability (curve $P(\phi_{k,attack}) = 100\%$), the attacker can derive a precision $\phi_{k,attack} = k\Delta_\phi$ for both algorithms as intended by the position sharing concept. With lower probability, the attacker can also gain a higher precision for OSPS-ASO as well as OSPS-FSO, so both algorithms show a certain degree of predictability. However, we can see that the predictability of OSPS-FSO is significantly lower (i.e., closer to the ideal curve $P = 100\%$) than for OSPS-ASO. For instance, with $P(\phi_{k,attack}) = 80\%$ probability and $k = 2$ compromised shares, the effective precision $\phi_{k,attack} = k$ of OSPS-ASO is 3.91 km and 4.58 km for OSPS-FSO (the ideal precision increase of a non-predictable share generation algorithm for $k = 2$ would be $\Delta_\phi = 10$ km $- 2 \cdot 2$ km $= 6$ km).

The fact that OSPS-FSO is less predictable than OSPS-ASO is also shown in Figure 9. Here, we consider the metric $P_{10\%}$ introduced above, i.e., the probability that an

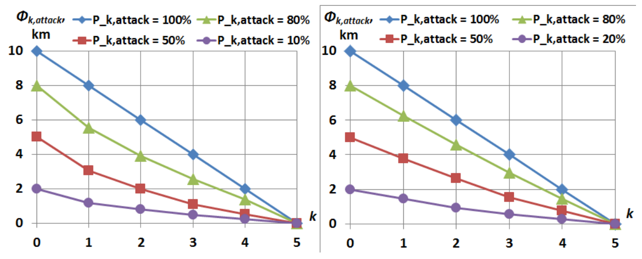


Fig. 8. (a) Precision $\phi_{k,attack}$ corresponding to probability values $P(\phi_{k,attack})$ depending on k for OSPS-ASO; (b) Precision $\phi_{k,attack}$ corresponding to probability values $P(\phi_{k,attack})$ depending on k for OSPS-FSO. $n = 5$; $r_0 = 10$ km; 100 runs of the Monte Carlo method

attacker can locate the user within an area of 10% size of the actually intended area resulting from the fusion of k shares. That is, instead of considering the absolute value of precision increase as before, we now consider the relative increase in precision. In the ideal case, $P_{10\%}$ should be 10%. This figure plots $P_{10\%}$ for OSPS-ASO and OSPS-FSO over different numbers of compromised shares out of a total number of $n = 5$ refinement shares ($\phi_{min} = 25$ km). Up to $k = 2$, both algorithms nearly lead to the same small increase in precision. However, for larger numbers of compromised shares, OSPS-FSO shows a much lower predictability than OSPS-ASO: for $k > 2$ its value of $P_{10\%}$ is more than 2 times lower.

So, overall we can state that for open space scenarios our new share generation algorithm OSPS-FSO creates shares which significantly more secure than created by OSPS-ASO.

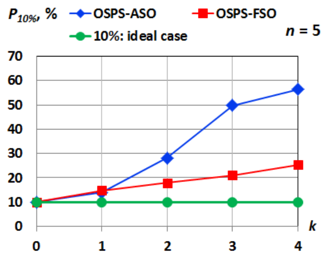


Fig. 9. Comparison of share generation algorithms: probability $P_{10\%}$ to derive an area covering 10% of the obfuscation circle c_k; $n = 5$; $r_0 = 25$ km; 100 runs using Monte Carlo method

4.2 Constrained Space Evaluation (OSPS-FSO vs. CSPS)

In this sub-section, we consider a constrained space movement model where MO only move in certain areas defined by the introduced map M_u. In this evaluation, we compare the proposed (open-space) approach OSPS-FSO, which does not consider any movement constraints, against the proposed map-based approach CSPS, which is aware of the movement constraints defined by M_u.

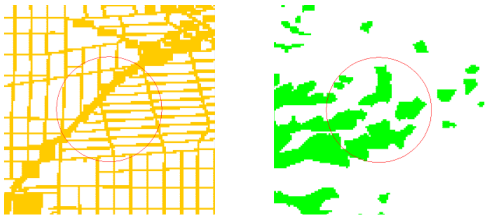


Fig. 10. (a) Map M_{LA}: roads and squares of Los Angeles City; (b) Map M_{BW}: forests of Baden Württemberg

Obviously, the difference between OSPS-FSO and CSPS depends on the concrete map. If the map does not define any constraints, OSPS-FSO and CSPS will behave similarly. If there are many constraints, we expect a bigger difference between both algorithms. Therefore, we used two real maps for our evaluation. The first map (M_{LA}) defines streets and places in a part of Los Angeles (see Figure 10a). The second map (M_{BW}) defines coarser movement constraints in a part of the state of Baden Württemberg (Germany), where the MO can move everywhere except for forests (see Figure 10b).

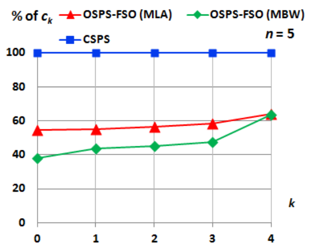


Fig. 11. Relative effective area size; 100 runs of the Monte Carlo method

In order to compare OSPS-FSO and CSPS, we compare the sizes of the obfuscation areas calculated by OSPS-FSO and CSPS after the intersection with the map M_u. CSPS adjusts the obfuscation area A_k such that the intersection area $A_k \cap M_u$ always has the desired obfuscation area size $\pi \cdot (\phi_{min} - k\Delta_\phi)^2$. In contrast, OSPS-FSO does not consider the map, which reduces the effective obfuscation area size where the MO can actually be located. Obviously, a smaller effective obfuscation area size is less secure. Therefore, as performance metric we calculate the *relative effective area size* of OSPS-FSO compared to the desired obfuscation area size $\pi \cdot (\phi_{min} - k\Delta_\phi)^2$. Since CSPS adjusts the obfuscation area such that the effective size is equal to $\pi \cdot (\phi_{min} - k\Delta_{\phi})^2$, it always has a relative size of 100%.

Figure 11 shows the relative effective area sizes for the two maps over different numbers of shares (k) known by the attacker. Each figure depicts the results for different total numbers (n) of shares. The curve labeled "OSPS-FSO (MLA)" depicts the results of OSPS-FSO for map M_{LA}; "OSPS-FSO (MBW)" depicts the results for map M_{BW}. First, we can observer that the coarse-grained map M_{BW} has a stronger effect on the effective area size since it constraints larger areas where the MO cannot be located. Moreover, we see that the relative effective area size increases for larger k. The reason for this is that usually smaller obfuscation areas (higher numbers k of refinement shares) tend to overlap more with regions where the user can actually be located. For instance, an area of only a few 10 meters will have almost 100% overlap with a building or street where the user can be located.

Comparing our map-aware approach CSPS and the open-space approach OSPS, we see that CSPS leads to a relative improvement of the effective obfuscation area size between about 40% and 60% for the two maps. Therefore, we can conclude that considering map knowledge is essential to guarantee the security of obfuscation. By considering map knowledge, CSPS guarantees that the effective obfuscation area size is equal to the desired area size for k shares.

5 Related Work

There are many different techniques to preserve the privacy of user locations while using LBS. They can be classified into methods based on access control, cryptographic encryption, k-anonymity, and spatial obfuscation ([6], [7], [8]).

The application of *access control* (e.g., [9]) using privacy policies allows users to define which LBAs are authorized to access the user's private location information. However, privacy policies do not provide ultimate (technical) guarantees against the misuse of user's data by LBS.

The classic method of *encryption* applied to the user's position information has a general drawback: no geometric operations over the encrypted data are possible at the LS, or only at a very high cost.

k-anonymity based methods (e.g., [10]) are managing the user position so that it cannot be distinguished from $k-1$ positions of other users. This is achieved by adding a trusted anonymizer to the system, which manages the interaction between users and LBS. The anonymizer updates the exact user position by a set of k user positions (k-cluster). The common problem of k-anonymity based approaches is that they require total trust in a third party that operates the anonymizer. Also, the needed cluster of k users is not always available, especially clusters which satisfy additional constraining parameters such as area size and position diversity.

Spatial *obfuscation* (e.g., [11]) secures the user position by sending coarsened location information to the LBS. There is no need for a trusted third party, but on the other hand the problem of trading-off between precision and privacy raises. By selecting a large obfuscation area a user makes it impossible to query his position with acceptable precision, while a fine-granular location information provides only a low privacy level.

Our position sharing approach presented in [1] removes this shortcoming of obfuscation approaches. It uses spatial obfuscation as a basic mechanism and allows for a gradual refinement of the user position's precision by collecting data shares from different providers. The idea of decomposing a user's position information into shares has been also applied by Marias et al. [12]. The proposed method based on *secret sharing* solves the problem of limited trust to providers, but provides no gradual refinement of precision: only the complete set of shares gives the target position, while the absence of even a single share results in the absence of position information of any precision.

The problem of map-awareness regarding obfuscation techniques is a relatively rare topic, although it affects significantly the spatial obfuscation approaches. The *PROBE* approach proposed by Damiani et al. [13] considers map features with pre-defined probability values assigned to them. The distribution of probability of a user to be located within the given region is also assumed to be known a-priori. Moreover, a personalized model of privacy sensitivity for various map features is presented. The algorithm expands the obfuscation cells over the discrete space step-by-step. The resulting obfuscation region can have any shape, but the approach lacks some flexibility due to the enforced cell-based space representation.

The *landscape-aware obfuscation* of Ardagna et al. [14] provides a defense against Bayesian inference based on the prior probability density over the given area. This work presents the theoretical background for map-awareness. Also Ardagna et al. present the idea of adjusting the radius of obfuscation disk in order to preserve the user's privacy affected by the landscape knowledge. In our work, we adopt the similar principle to the position sharing approach.

6 Summary

In this paper, we have presented a new position sharing approach for managing obfuscated user positions on a set of untrusted location services (LSs). The basic idea of position sharing is to split the precise user position into a set of imprecise position shares and distribute these shares among LSs of different providers. Location-based applications (LBA) can query these shares from the LSs and fuse them to a position of well-defined precision depending on the number of shares they got access rights for from the tracked user. Since each LS only stores a single share of well-defined precision, a compromised LS will not reveal the precise user position but rather a position of strictly limited precision (graceful degradation of privacy).

We have presented enhanced share generation algorithms and fusion algorithms that further improve our basic position sharing approach presented in [1]. Firstly, we reduced the predictability of share generation that allows an attacker to gain further information from a set of compromised shares to further increase the position precision. Secondly, we presented a position sharing algorithm for constrained movement scenarios. Since typically users are likely to be located in areas like streets, buildings, etc. rather than areas like lakes or agriculture fields, open space approaches such as [1] are vulnerable to map-based attacks. Therefore, we presented a share generation algorithm that takes map knowledge into account.

Possible future research directions include the following. CSPS could be modified to additionally provide k-anonymity guarantees by adjusting the obfuscation area until it covers k other users. Moreover, the current approach is targeted at "position check-in" scenarios where the user only sporadically updates his position rather than continuously sending position updates. In future work we plan to design position sharing approaches that also support such continuous tracking scenarios. Finally, it would be interesting to also consider individual trust-levels of different location service providers. For instance, we could store more shares or shares of higher precision on LSs that are more trustworthy than others.

References

1. F. Dürr, P. Skvortsov, and K. Rothermel, "Position sharing for location privacy in non-trusted systems," in *PerCom 2011*, Seattle, USA, March 2011, pp. 189–196.
2. Privacy Rights Clearinghouse, "Privacy rights clearinghouse," http://www.privacyrights.org/data-breach, June 2011.
3. M. F. Mokbel, "Privacy in location-based services: State-of-the-art and research directions," in *MDM'07*, Mannheim, Germany, May 2007.
4. D. Pedreschi, F. Bonchi, F. Turini, V. S. Verykios, M. Atzori, B. Malin, B. Moelans, and Y. Saygin, "Privacy protection: Regulations and technologies, opportunities and threats," *Mobility, Data Mining and Privacy*, pp. 101–119, 2008.
5. A. Gutscher, "A Trust Model for an Open, Decentralized Reputation System," in *IFIPTM'07*, August 2007.
6. D. Riboni, L. Pareschi, and C. Bettini, "Privacy in georeferenced context-aware services: A survey," in *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications*, October 2008.
7. J. Krumm, "A survey of computational location privacy," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, August 2009.
8. A. Solanas, J. Domingo-Ferrer, and A. Martínez-Ballesté, "Location privacy in location-based services: Beyond ttp-based schemes," in *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications (PiLBA)*, Malaga, Spain, October 2008.
9. C. Hauser and M. Kabatnik, "Towards privacy support in a global location service," in *WATM/EUNICE'01*, September 2001.
10. P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 12, pp. 1719–1733, Dec. 2007.
11. C. Ardagna, M. Cremonini, E. Damiani, S. De Capitani di Vimercati, and P. Samarati, "Location privacy protection through obfuscation-based techniques," in *Proc. of the 21st IFIP WG 11.3 Working Conference on Data and Applications Security*, vol. 4602, 2007.
12. G. F. Marias, C. Delakouridis, L. Kazatzopoulos, and P. Georgiadis, "Location privacy through secret sharing techniques," in *WOWMOM'05*. IEEE Computer Society, June 2005.
13. M. L. Damiani, E. Bertino, and C. Silvestri, "Protecting location privacy against spatial inferences: the probe approach," in *SIGSPATIAL ACM GIS 2009 Intl. Workshop on Security and Privacy in GIS and LBS*, ser. SPRINGL '09. New York, USA: ACM, 2009.
14. C. A. Ardagna, M. Cremonini, and G. Gianini, "Landscape-aware location-privacy protection in location-based services," *Journal of System Architecture (JSA)*, vol. 55, April 2009.