PShare: Ensuring Location Privacy in Non-trusted Systems through Multi-Secret Sharing

Marius Wernke, Frank Dürr, Kurt Rothermel

Institute of Parallel and Distributed Systems, University of Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

Abstract

Location-based applications such as Facebook Places, Foursquare, or Loopt typically use location services to manage mobile object positions. However, exposing precise user positions raises user privacy concerns, especially if location service providers are not fully trusted. To enable the secure management of private user positions in non-trusted systems, we present two novel position sharing approaches based on the concept of multi-secret sharing. We improve existing geometric position sharing approaches [1, 2] by considering continuous position updates and by increasing the robustness against various attacks. Furthermore, we present the first position sharing approach for symbolic location models.

Keywords: Location-based applications, privacy, position sharing, location management

1. Introduction

Driven by the rapid spread of mobile devices integrating positioning sensors, so-called location-based applications (LBAs) attract millions of users today. Typical examples of those applications include point of interest finders (e.g., Qype), friend finders (e.g., Loopt), or pay as you drive insurances (e.g., PAYD). Another prominent class of LBAs is geosocial networking, such as Facebook Places, Foursquare, or Gowalla, allowing users for "checking in" at different locations to share their current position with friends.

LBAs typically make use of so-called location services, which manage mobile object positions and allow for position sharing between the users of one or more applications. Mobile objects inform the corresponding location service about their current position, while clients of this service can query location information, e.g., by means of position, range, or nearest neighbor queries. An efficient and effective location service is a prerequisite for most of today's LBAs, which might be either an "LBA internal" or a public location service as already offered today in the Internet, such as Google Latitude, Yahoo Fire Eagle, or Trace4You.

While sharing of location information is a highly desirable feature from an application's point of view, it gives rise to severe privacy concerns. Therefore, location services typically provide access control mechanisms allowing users whose location information is managed by the service to define who can access their location information in which granularity. Most of these mechanisms assume that the location service is fully trusted, and hence will ensure that location information is only exposed to legitimate users. Unfortunately, many cases in the past have shown that private user information has been "lost" or leaked by service providers that were supposed to be trustworthy [3]. As a consequence, assuming a location service to be fully trusted is at least questionable.

Therefore, several approaches for the protection of position information in the absence of trusted parties have been proposed. The simplest approach is to encrypt position information before sending it from the mobile object to the location service. However, this approach prevents server-side processing of position information, which is needed for most queries, such as range or nearest neighbor queries. Another rather simple but limited approach is called

Email address: <Marius.Wernke|Frank.Duerr|Kurt.Rothermel>@ipvs.uni-stuttgart.de (Marius Wernke, Frank Dürr, Kurt Rothermel)

NOTICE: this is the author's version of a work that was accepted for publication in Pervasive and Mobile Computing. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in M. Wernke, et al., PShare: Ensuring location privacy in non-trusted systems through multi-secret sharing, Pervasive and Mobile Computing (2013), http://dx.doi.org/10.1016/j.pmcj.2013.01.001

position obfuscation. Here, the position information is deliberately degraded before it is sent to the location service. Therefore, an attacker (including a compromised location service) will only see the degraded position rather than the precise user position. The obvious limitation of this approach is that the trustworthiness of the location service determines the precision of the location data provided to the application, i.e., it may reduce both the spectrum of possible applications and the quality of applications substantially. To overcome this problem, we proposed a concept that combines obfuscation with *position sharing* [1, 2]. The basic idea of this concept is to let the mobile object split its precise position into a set of *position shares* of strictly limited precision. These shares are distributed to *multiple* location services offered by different providers. Therefore, as in the pure obfuscation approach described above, a compromised location service can only reveal information of limited precision. However, the precision of position information can be incrementally increased by combining shares. In fact, the obfuscation can be undone by combining all shares, i.e., the original precision as captured by the position system can be restored. By allowing mobile objects to control the set of shares accessible by a particular client, different precision levels can be provided to different clients, ranging from the lowest level up to the original precision. Another advantage of this scheme is that it provides graceful degradation of privacy in the presence of compromised location services: The precision of the revealed position information only increases with the number of compromised location services.

While the position sharing method sketched above applies geometric transformations for obfuscation, the scheme presented in this paper is based on the concept of multi-secret sharing [4]. Using multi-secret sharing for position sharing improves our previous scheme in several ways: First, it can be applied not only to geometric positions (longitude, latitude values) but also to *symbolic locations*, such as cities, buildings, or restaurants, which are important for a wide range of applications. Second, by using multi-secret sharing, which is based on modular arithmetic rather than probabilistic geometric transformations, we improve the robustness of the scheme significantly. Our previous scheme is subject to probabilistic attacks, where an attacker tries to compute the probability distribution function of positions to increase the precision. In contrast, our deterministic obfuscation approach used in this paper makes such attacks impossible. Third, the novel scheme not only considers isolated position check-ins, but also subsequent position updates of the same mobile object, which might unintentionally increase precision if performed in an uncontrolled manner. Finally, we take map knowledge into account to provide counter-measures against attackers using map matching to increase precision.

The rest of this paper is structured as follows. Next, we present related work in Section 2. In Section 3, we introduce our system model. The two variants of our scheme for geometric and symbolic locations are described in Sections 4 and 5. In Section 6, we present an extension for geometric locations taking map knowledge into account. Then, we analyze the robustness against various attacks in Section 7. In Section 8, we present an evaluation using real world traces to show the applicability of our approaches. A performance evaluation is presented in Section 9, and finally, we conclude with a summary and outlook on future work.

2. Related Work

The most prominent location privacy concept is *k-anonymity* [5], which protects user identities by guaranteeing that the user is indistinguishable from at least $k-1$ other users. However, *k-anonymity* approaches and extensions such as *l-diversity* [6] or *t-closeness* [7] usually require a trusted third party anonymizer. In contrast, we aim for approaches protecting privacy without a trusted third party.

Spatial obfuscation protects user positions by decreasing their precision [8, 9, 10]. As already pointed out, these approaches can be implemented without a trusted third party. However, the precision offered to clients is limited by the precision of positions stored at the location service. In contrast, we allow for different precision levels for clients.

Dummy approaches send the true user position together with several false positions to the location service [11]. However, dummy identification can reduce their effectiveness. Thus, more advanced approaches like [12] make dummy identification more difficult using databases of movement trajectories. However, this leads to the problem of collecting trajectories without raising privacy concerns, and of operating such a database without a trusted third party so it cannot be manipulated.

Our previous position sharing approach [1] and its extension to maps [2] can be used without a trusted third party. These approaches split up a precise user position into several position shares of limited precision that are distributed to different location services. Clients can reconstruct the position in different granularities by combining several shares

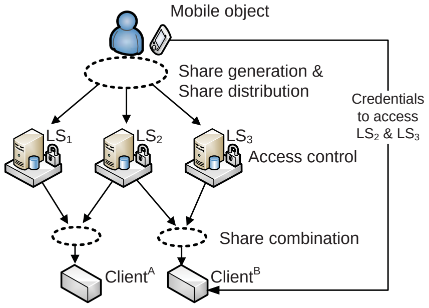


Figure 1: System components

from different services. As already pointed out, our new approach uses a different technique (multi-secret sharing) to improve our previous approaches w.r.t. supported location models (geometric & symbolic), robustness, and by considering position updates.

An approach based on (single) secret sharing was presented in [13]. It divides the user position into several shares so that only a predefined number of shares can reconstruct the user position. This approach provides user privacy, however, it reveals the precise user position to each client. In contrast, our approaches ensure *graceful degradation of privacy* instead of implementing an "all-or-nothing" approach. Therefore, clients can be assigned different precision levels without revealing the precise user position.

3. System Model and Requirements

The system model consists of three different components as presented in Fig. 1.

The **mobile object** (MO) uses an integrated positioning system, such as GPS, to determine the precise current MO's position π. We assume that the MO is not compromised and no malicious software component can access π. The only component allowed to directly access the integrated positioning system is our share generation component (see below). All other components on the MO query the MO's position as shown below. In order to ensure that no other component than the share generator can directly access the integrated positioning system, trusted computing approaches can be used, based for instance on trusted module hardware as presented in [14]. For a detailed description of these approaches, we refer to [14]. The MO executes a local software component for share generation that splits up π into a *master share* m_π, denoted as *m*-share, and set $S_\pi = \{r_{\pi,1}, \ldots, r_{\pi,n}\}$ of *n refinement shares*, denoted as *r*-shares, by calculating

$$generate(\pi, l_{max}, n) = (m_\pi, S_\pi).$$

Parameter l_{max} defines the number of different *precision levels*, i.e., positions of different well-defined precisions that can be offered to clients. We use the notation $p(\pi, l)$ to denote a position on precision level l derived from the precise position π. $p(\pi, 0)$ represents the least precise position on level 0, and $p(\pi, l_{max})$ the position of highest precision on level l_{max}. The concrete definition of precision is dependent on the type of location model (geometric or symbolic), and is introduced later for each model.

The *m*-share m_π consists of position $p(\pi, 0)$ and additional information required to reconstruct positions of higher precision levels greater 0. m_π is public, i.e., everyone knows the least precise position $p(\pi, 0)$. *r*-shares contain further secret information to refine $p(\pi, 0)$ to more precise positions of higher levels (see below). After share generation, the *r*-shares are distributed to different location servers such that each server receives one *r*-share.

Location servers (LSs) store and manage *r*-shares. Each LS implements an access control mechanism for *r*-shares. The access rights are defined by the MO and provided to the clients of the LS, for instance, as credentials to access a certain number of *r*-shares, where the number of accessible shares defines the intended precision offered to a client.

Clients receive permissions to access a well-defined set $S'_\pi \subseteq S_\pi$ of r-shares from the MO and perform the *share combination* on the public m-share and these r-shares:

$$combine(m_\pi, S'_\pi) = p(\pi, l)$$

Combining the m-share m_π and the set S'_π of r-shares yields position $p(\pi, l)$ of precision level l. Local clients running on the mobile device directly access set $S'_\pi \subseteq S_\pi$ of r-shares provided by the share generation component to reconstruct $p(\pi, l)$. The concepts for share combination are identical to the case where shares are queried from remote LSs. Therefore, we focus the following descriptions on the more general case, where clients access different r-shares from a set of remote LSs.

The goal is now to design *secure* share generation and combination algorithms such that an attacker—either (malicious) client or LS—knowing a set S'_π of shares refining $p(\pi, l)$ of precision level l cannot derive a position of higher precision than $p(\pi, l)$. Formally, the condition

$$precision(p(\pi, l)) \geq precision(\pi_{attack})$$

must hold, where $precision(\pi_{attack})$ defines the precision of position π_{attack} calculated by the attacker. This is the essential requirement for our approach. Otherwise, the MO could not control the precision offered to LSs and clients by granting access to a certain number of shares.

Note that the basic assumption of position sharing is that the unauthorized access of shares cannot be prevented perfectly. Thus, a malicious client or LS could get a position of the precision defined by the accessible shares with a certain probability. However, using secure shares, we can limit this risk by limiting the precision that can be derived from a certain number of (compromised) shares.

4. *PShare-GLM*: Geometric Position Sharing

We start the description of our position sharing approaches with *PShare-GLM*, the approach for geometric location models. First, we introduce our geometric location model, which is used to define positions on different precision levels, and give an overview on how to apply multi-secret sharing to position sharing. Then, we describe the algorithmic details of share generation and combination.

4.1. Geometric Location Model

In *PShare-GLM*, the precise MO position π and obfuscated positions $p(\pi, l)$ are defined as geometric locations based on a Cartesian coordinate system. We use a common map projection, e.g., Universal Transverse Mercator (UTM) projection, to map ellipsoidal coordinates (longitude, latitude) to Cartesian coordinates. The UTM projection divides the Earth into sixty zones, each representing a six degree band of longitude. For an MO traveling from one zone to another, the zone is changed as soon as the area of the new check-in location is completely covered by the new zone. Position π is a point coordinate. A position $p(\pi, l)$ of precision level l is defined as square area $p(\pi, l) = ((x_l, y_l), b^{l_{max}-l})$, where (x_l, y_l) defines the coordinates of the south-west corner of the square, and $b^{l_{max}-l}$ the side length. Hence, the *precision* corresponds to the side length of the square. Parameter b defines the granularity of the precision levels, where an increase of the precision level by 1 increases the precision by a factor of b and partitions the area of $p(\pi, l)$ into b^2 squares. For $b = 2$ the result is a quadtree as depicted in Fig. 2, where each position of level l is refined into 4 positions of level $l + 1$.

To encode a position on level l, we specify the x and y coordinates of $p(\pi, l)$ as n digits with base b:

$$\pi.x = \sum_{k=0}^{n-1} \alpha_k b^k = (\alpha_{n-1} \cdots \alpha_1 \alpha_0)_b$$

$$\pi.y = \sum_{k=0}^{n-1} \beta_k b^k = (\beta_{n-1} \cdots \beta_1 \beta_0)_b$$

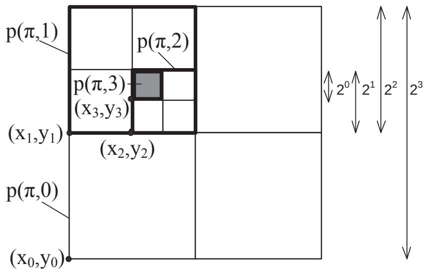


Figure 2: Geometric area of $p(\pi, l)$ for $b = 2$ and $l_{max} = 3$

Position π is degraded to $p(\pi, l)$ by setting the $l_{max} - l$ least significant digits to 0, meaning the actual digit values are unknown. For instance, for $b = 2$ and $l_{max} = 3$, $p(\pi, 0)$ can be written as follows, where underlined digits are unknown:

$$p(\pi, 0).x = 00010101011101110011\underline{000}$$
$$p(\pi, 0).y = 11100100110001000101\underline{000}$$

4.2. *Overview of Multi-Secret Sharing Algorithm*

PShare-GLM utilizes multi-secret sharing algorithms for share generation and combination. Therefore, we first give a brief introduction to multi-secret sharing, before we describe the basic relation between multi-secret and position sharing.

Multi-secret sharing is an extension of *secret sharing*. A widely known secret sharing scheme is Shamir's (t, n)-threshold scheme [15]. The general idea of this scheme is to split up a secret, say K, into a set of n *shares* that can be distributed to different participants. The so-called *dealer*, which initiates secret sharing, defines a threshold value t, which defines the required number of shares to reconstruct K, and distributes the shares to the participants, where each participant owns one share. Any t out of the n participants putting their shares together can reconstruct secret K. If less than t shares are available, K cannot be reconstructed.

The general idea of *multi-secret sharing* is that a dealer splits up m secrets $K_1, \ldots, K_m$ into a set of n shares so that each secret K_i can be reconstructed by any set of at least $t_i \leq n$ shares. The number t_i of required shares to reconstruct each secret K_i is again defined by the dealer. For less than t_i shares, no information about K_i is exposed.

We apply the idea of multi-secret sharing as follows to our position sharing approach *PShare-GLM*. We use the positions $p(\pi, 1), \ldots, p(\pi, l_{max})$ as secrets $K_1, \ldots, K_{l_{max}}$ of the multi-secret sharing scheme. The MO corresponds to the "dealer", which creates n r-shares using function *generate*(π, l_{max}, n) as presented in the next subsection in detail. We assign each precision level l the threshold value $t_l = l$, i.e., l shares are required to reveal $p(\pi, l)$. However, our approach provides the flexibility to use any number t_l for level l, where greater values increase robustness at the price of a greater overhead as discussed later.

The r-shares are then distributed among n LSs by the MO. The role of "participants" is split up between LSs and clients. Whereas participants of the original multi-secret sharing scheme manage *and* combine shares, LSs only manage at most one share per position, and clients combine multiple shares queried from different LSs. This role split allows for providing different precision levels to different clients, and limits the precision known by a single LS.

The m-share contains, similar to traditional multi-secret sharing, public data necessary for share combination (see next subsection). However, in contrast to multi-secret sharing, the m-share additionally contains coarse-grained position information serving as origin for the refinements.

4.3. *Share Generation*

The following description of share generation is based on the multi-secret sharing approach of Chan et al. [4]. However, also other multi-secret sharing approaches could be applied.

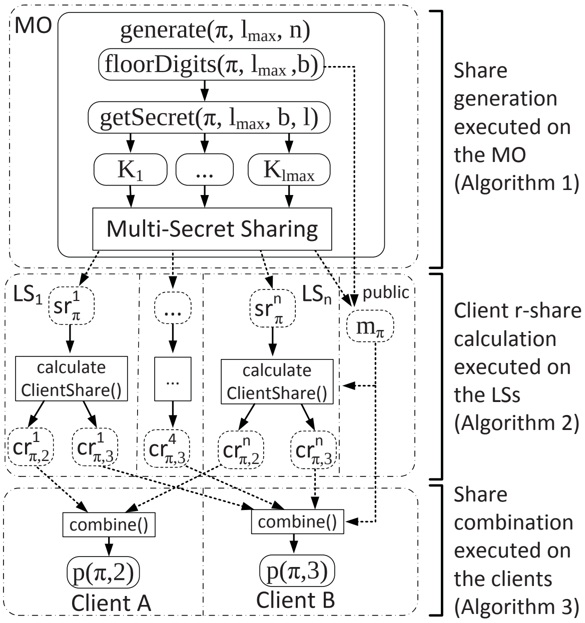


Figure 3: *PShare-GLM* process overview

Figure 3 visualizes the whole process of share generation and combination. Algorithm 1 defines the process of share generation, which is entirely performed by the MO. After sensing π, the MO first calculates $p(\pi, 0)$ (position of minimal precision) by simply setting the least l_{max} significant digits to zero as described in Section 4.1 using function *floorDigits*(π, l_{max}, b). $p(\pi, 0)$ is part of the public m-share, which is denoted as m_π.

Then, the MO calculates the r-shares for each precision level greater zero. As already mentioned in the previous subsection, the basic idea is to create one secret of the multi-secret scheme for each position $p(\pi, l)$ with $l > 0$. To this end, we first have to translate each position $p(\pi, l)$ into secret K_l using function *getSecret*(π, l_{max}, b, l). Since K_l is a single integer number, x and y values are to be encoded as a single number. This is done by interleaving the digits of x and y values. In more detail, function *getSecret*(π, l_{max}, b, l) calculates each secret K_l as difference of the interleaved digit values of $p(\pi, l_{max})$ and $p(\pi, 0)$ with the $2 * (l_{max} - l)$ least significant digits set to 0.

After the translation of $p(\pi, l)$ to secret K_l, Chan's multi-secret sharing scheme is applied to the calculated secrets $K_1, \ldots, K_{l_{max}}$. To protect a secret, a secret polynomial $f_l(X)$ of degree $t_l - 1$ is calculated by the MO for each secret K_l:

$$f_l(X) = a'_0 + a'_1 X + \cdots + a'_{t_l-1} X^{t_l-1}$$

The constant term a'_0 corresponds to the protected secret K_l. The polynomial and therefore the secret can be reconstructed by polynomial interpolation using modular arithmetic if t_l distinct points of it are known. Therefore, each r-share contains information to determine a single distinct point (x, y) of the secret polynomial as shown below.

According to the multi-secret scheme, the secret polynomials $f_1(X), \ldots, f_{l_{max}}(X)$ of all secrets are packed together using the *Chinese Remainder Theorem* into one single secret polynomial $f(X)$:

$$f(X) = a_0 + a_1 X + \cdots + a_{t_{l_{max}}-1} X^{t_{l_{max}}-1}$$

$f(X)$ is defined, such that $f_l(X) \equiv f(X) \bmod p_l$. That is, we can calculate $f_l(X)$ by calculating $f(X)$ modulo p_l for a prime p_l defining the field $\mathbb{Z}_{p_l}[X]$ of $f_l(X)$. $f_l(X)$ is a uniquely defined polynomial of degree equal to or less than $t_l - 1$ over $\mathbb{Z}_{p_l}[X]$ with $a_j \equiv 0 \bmod p_k$ for all coefficients a_j with $j \geq t_l$ and $k = 1, 2, \ldots, l - 1$. The set of primes $P = \{p_1, \ldots, p_{l_{max}}\}$, which is required together with the r-shares to reconstruct the secrets, is part of the m-share.

It remains the question, which information is contained in an r-share in detail. As pointed out above, each r-share should contain information about a single distinct point (x, y) of a certain polynomial $f_l(X)$. Using multi-secret

Algorithm 1 *PShare-GLM*: Share generation (MO)

```
Function: $generate(\pi, l_{max}, n)$
$p(\pi, 0) \leftarrow floorDigits(\pi, l_{max}, b)$
for $l = 1$ to $l_{max}$ do
    $K_l \leftarrow getSecret(\pi, l_{max}, b, l)$
end for
$P, f(X) \leftarrow calculatePolynomial(K)$
$X \leftarrow n$ distinct integer
for all $x_i \in X$ do
    $y'_i \leftarrow f(x_i)$
    $sr^i_\pi \leftarrow (x_i, y'_i)$
end for
$m_\pi \leftarrow P, p(\pi, 0)$
return $m_\pi, \{sr^1_\pi, \ldots, sr^n_\pi\}$
```

Algorithm 2 *PShare-GLM*: Client r-share calculation (LS)

```
Function: $calculateClientShare(l)$
$cr^{LS}_{\pi,l}.x \leftarrow sr^{LS}_\pi.x$
$cr^{LS}_{\pi,l}.y \leftarrow sr^{LS}_\pi.y'$ mod $m_\pi.p_l$
return $cr^{LS}_{\pi,l}$
```

sharing, we actually have to distinguish between the information of the r-shares generated by the MO, which is sent to and stored by the LSs (called *server r-share* sr^{LS}_π), and the information sent from the LSs to the clients (called *client r-share* $cr^{LS}_{\pi,l}$). Each server r-share contains a distinct point (x, y') of the secret polynomial $f(X)$. Each client r-share contains a distinct point (x, y) of $f_l(X)$, which is required for share combination. $cr^{LS}_{\pi,l}$ is calculated by the LS from sr^{LS}_π as $y = y'$ mod p_l upon a request of the client for $cr^{LS}_{\pi,l}$ (cf. Alg. 2). Note that different client r-shares of different levels can be calculated from one server r-share using the specific prime of level l. In the next subsection, we describe the details of share combination.

4.4. Share Combination

In order to calculate $p(\pi, l)$, a client retrieves the publicly available m-share m_π and t_l client r-shares $cr^{LS_1}_{\pi,l}, \ldots, cr^{LS_{t_l}}_{\pi,l}$ from t_l different LSs. As described in the previous subsection, m_π contains the set of prime numbers (P) and the least precise position $p(\pi, 0)$ of the MO; each client r-share $cr^{LS_i}_{\pi,l}$ defines a distinct point (x_i, y_i) in $f_l(X)$.

Share combination (Alg. 3) uses the *Lagrange interpolation* over the field $\mathbb{Z}_{p_l}[X]$ (line 1). It reconstructs polynomial $f_l(X)$ by interpolating the t_l distinct points of the client r-shares, which uniquely define $f_l(X)$. Secret K_l is the constant term of $f_l(X)$ and is calculated as $f_l(0) \equiv K_l \mod p_l$. Position $p(\pi, l)$ is calculated by adding the reconstructed secret K_l to the interleaved representation of $p(\pi, 0)$ and splitting up the sum into the x and y values of $p(\pi, l)$ (line 3).

Each polynomial $f_l(X) \in \mathbb{Z}_{p_l}[X]$ has a degree of at most $t_l - 1$ and fulfills the condition $f_l(x_i) = y_i$. Because at least t_l distinct points are required to interpolate a polynomial of degree $t_l - 1$, it is guaranteed that $p(\pi, l)$ cannot be reconstructed with less than t_l client r-shares.

4.5. Multiple Position Updates

Up to now, we only considered share generation for single position updates. However, in the worst case a compromised LS or client could reveal a complete history of positions for a certain precision level (either precision level 1 for an LS with access to a single server r-share, or a certain level l in case of a client that was granted access to the client r-shares of level l). From the literature, it is well known that the knowledge of multiple obfuscated positions might enable attackers to further refine the precision beyond the intended precision [9]. To avoid this, we now present an extension of our basic algorithm.

Algorithm 3 *PShare-GLM*: Share combination (client)

```
Function: $combine(m_\pi, \{cr_{\pi,l}^{LS_1}, \dots, cr_{\pi,l}^{LS_{t_l}}\}, l)$
1: $f_l(X) \leftarrow Lagrange(\{cr_{\pi,l}^{LS_1}, \dots, cr_{\pi,l}^{LS_{t_l}}\}, m_\pi.p_l)$
2: $K_l \leftarrow f_l(0)$
3: $p(\pi, l) \leftarrow split(interleave(m_\pi.p(\pi, 0)) + K_l)$
4: return $p(\pi, l)$
```

We assume that the MO has a known maximum velocity v_{max}, which is also known by an attacker. Moreover, we consider the fact that in the worst case an attacker knows the complete history $U = \{(p(\pi_{first}, l), t_{first}), \dots, (p(\pi_{last}, l), t_{last})\}$ of position updates for a certain precision level l. Here, t_{first} denotes the time of the first update, and t_{last} the time of the last update up to the present time. Level l depends on the available shares accessible by the attacker. Then, we have to guarantee that for all $t \in [t_{first}, t_{last}]$ the attacker cannot derive a position of higher precision than the precision of $p(\pi_t, l)$.

Before describing our counter measure, we have to consider the so-called *maximum velocity attack* [9] in more detail. For this attack, the attacker considers two succeeding positions $(p(\pi_i, l), t_i)$ and $(p(\pi_{i+1}, l), t_{i+1})$ with the obfuscated areas $A = p(\pi_i, l)$ and $B = p(\pi_{i+1}, l)$ at times $t_A = t_i$ and $t_B = t_{i+1}$. In [9], it has been shown that a sequence of position updates resist a maximum velocity attack, if each pair of succeeding updates resists a maximum velocity attack. Therefore, it is sufficient to only consider two directly succeeding positions. With this information, the attacker tries to remove areas from B that are not reachable from A in time $\delta t = |t_B - t_A|$ for an MO traveling with speed v_{max}. Furthermore, the attacker tries to remove areas from A without reachable point in B considering δt and v_{max}. By removing unreachable parts of A or B, the obfuscation area is decreased and the precision of the attacker is increased.

To prevent such attacks, we first only consider position updates for level 0. Later we will show that protecting the position updates of level 0 against maximum velocity attacks also protects the position updates for every level $0 \leq l \leq l_{max}$. Let $d_{pp}(p(\pi_i, 0), p(\pi_{i+1}, 0))$ be the point pairwise distance of two succeeding MO positions $p(\pi_i, 0)$ and $p(\pi_{i+1}, 0)$. The point pairwise distance of two rectangular areas is defined as the maximum Euclidean distance between any point in the first area to any point in the second area. Furthermore, let $\delta t = |t_{i+1} - t_i|$ be the time between the two updates. Then the MO only sends an update for $p(\pi_{i+1}, 0)$ at time t_{i+1}, if the following condition is fulfilled:

$$\delta t \geq \frac{d_{pp}(p(\pi_i, 0), p(\pi_{i+1}, 0))}{v_{max}}. \tag{1}$$

If this condition is fulfilled, every point in $p(\pi_{i+1}, 0)$ is reachable from $p(\pi_i, 0)$ in time δt. Otherwise, $p(\pi_{i+1}, 0)$ has to be delayed until this condition is fulfilled.

For precision levels greater than zero, the point pairwise distance is always smaller than or equal to the point pairwise distance of level 0. Thus, if Equ. 1 is fulfilled for level 0, it is also fulfilled for levels greater than 0.

The *maximum delay time* Δt between two updates depends on the values of b and l_{max} and is defined as:

$$\Delta t = \frac{2 * (\sqrt{2} * b^{l_{max}})}{v_{max}}. \tag{2}$$

Intuitively, Δt describes the maximum time that is required to travel a distance of two times the diagonal of the area of $p(\pi, 0)$ with v_{max}. Therefore, the MO can trade-off the maximum delay time against the minimal revealed precision ($p(\pi, 0)$) by adjusting the size of $p(\pi, 0)$.

We analyzed the check-in behavior of users from different location-based applications in our real world evaluation presented in Section 8 and show that only few check-ins have to be delayed. For applications requiring higher update rates than used for sporadic check-ins, the *maximum precision decrease* Δs introduced by the maximum delay time Δt is defined by the distance of two times the diagonal of the area of $p(\pi, 0)$:

$$\Delta s = 2 * \sqrt{2} * b^{l_{max}}. \tag{3}$$

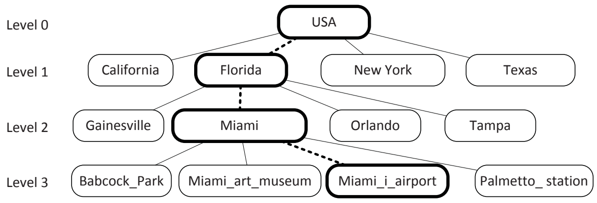


Figure 4: Simplified location hierarchy example

5. *PShare-SLM*: Symbolic Position Sharing

Next, we present *PShare-SLM*, the symbolic counterpart to the position sharing algorithm *PShare-GLM*. The general idea of *PShare-SLM* is similar to *PShare-GLM*: We apply the multi-secret sharing scheme [4] to share the MO's position in different precision levels with different clients. Since the symbolic location definition differs from the geometric definition, we start with an explanation of our symbolic location model, before we present the specific share generation and combination algorithms.

5.1. Symbolic Location Model

Our symbolic location model consists of a location hierarchy based on the spatial contains relationship. Each location is represented as a vertex v in the hierarchy, and has a level l defining the length of the path from the root to v (cf. Fig. 4). The root location is on level 0, and we assume that all leaf vertices have the same level l_{max}. Each location has a unique location name in the context of its parent location, for instance, "Florida" for the location representing the State of Florida as child of the location representing the country USA. The concatenation of names on the path from the root to a location defines a unique label for each location, such as `usa/florida/miami/miami_i_airport` for the location representing the Miami International Airport. Each location label can be mapped to a unique identifier represented as integer, which serves as input to the secret sharing scheme as presented below.

Using a hierarchical model makes it easy to define positions of different precisions. Each hierarchy level defines a precision level, where level l_{max} defines the highest precision where the MO is located. Again, $p(\pi, l)$ denotes a position of precision level l similar to the geometric model. However, $p(\pi, l)$ is now represented as symbolic location identifier rather than geometric coordinates. The sequence of ancestor vertices of a position $p(\pi, l)$ is denoted as

$$ancestors(p(\pi, l)) = (p(\pi, 0), \ldots, p(\pi, l)).$$

5.2. Share Generation and Share Combination

We now apply the idea of multi-secret sharing to our symbolic position sharing approach *PShare-SLM*. The share generation executed by the MO is shown in Alg. 4.

First, the MO calculates $ancestors(p(\pi, l_{max}))$ to determine $(p(\pi, 0), p(\pi, 1), \ldots, p(\pi, l_{max}))$ (line 1). The identifier of $p(\pi, 0)$ defines the root of the hierarchy and is stored in the m-share. The identifiers of $p(\pi, 1)$ to $p(\pi, l_{max})$ are used as the secrets $K_1, \ldots, K_{l_{max}}$ for the multi-secret sharing scheme. The remaining part of the algorithm is similar to the share generation of *PShare-GLM*. That is, we apply the multi-secret sharing algorithm by calculating the server r-shares, and distribute them to the LSs.

Similarly, the share combination algorithm as depicted in Alg. 5 uses again the *Lagrange interpolation* over the field $\mathbb{Z}_{p_l}[X]$ to reconstruct the secret polynomial $f_l(X)$ for a given level l. The constant term of $f_l(X)$ is the identifier of $p(\pi, l)$, which can be mapped to the label of $p(\pi, l)$.

Algorithm 4 *PShare-SLM*: Share generation (MO)

```
Function: $generate(\pi, l_{max}, n)$
$p(\pi, 0), \ldots, p(\pi, l_{max}) \leftarrow ancestors(p(\pi, l_{max}))$
$K \leftarrow p(\pi, 1).id, \ldots, p(\pi, l_{max}).id$
$P, f(X) \leftarrow calculatePolynomial(K)$
$X \leftarrow n$ distinct integer
for all $x_i \in X$ do
    $y'_i \leftarrow f(x_i)$
    $sr_\pi^i \leftarrow (x_i, y'_i)$
end for
$m_\pi \leftarrow P, p(\pi, 0).id$
return $m_\pi, \{sr_\pi^1, \ldots, sr_\pi^n\}$
```

Algorithm 5 *PShare-SLM*: Share combination (client)

```
Function: $combine(m_\pi, \{cr_{\pi,l}^{LS_1}, \ldots, cr_{\pi,l}^{LS_{t_l}}\}, l)$
$f_l(X) \leftarrow Lagrange(\{cr_{\pi,l}^{LS_1}, \ldots, cr_{\pi,l}^{LS_{t_l}}\}, m_\pi.p_l)$
$p(\pi, l).id \leftarrow f_l(0)$
return $p(\pi, l)$
```

5.3. Multiple Position Updates

Next, we analyze *PShare-SLM* with regard to multiple symbolic position updates. As pointed out above, an attacker knowing the complete position history $U = \{(p(\pi_{first}, l), t_{first}), \ldots, (p(\pi_{last}, l), t_{last})\}$ of a certain level l could try to use a maximum velocity attack to increase precision. Note that although the location hierarchy itself does not define the distance information necessary for such attacks, an attacker can determine this information by matching symbolic locations to available topographic maps.

The basic idea to counter such attacks is similar to the geometric case: A new update $p(\pi_{i+1}, l)$ is only permitted if any position within $p(\pi_{i+1}, 0)$ is reachable from any position within $p(\pi_i, 0)$ considering $\delta t = |t_{i+1} - t_i|$ and the MO's maximum velocity v_{max}. Although theoretically this would be an effective counter measure, it impacts the minimal update time between two succeeding updates as specified in Equ. 1. In contrast to the geometric case, where the precision of $p(\pi_i, 0)$ can be specified by the MO, level 0 is now defined by the root location of the given symbolic location model. Therefore, it is worthwhile to have a closer look at the influence on the minimal update time.

Assume a model where the root location covers a whole country like Germany. In this case, the maximum distance between two positions within the hierarchy would be about 1 000 km. For a MO walking with $v_{max} = 6\,km/h$ this results in a maximum delay of 6.94 days, whereas a maximum velocity of $v_{max} = 200\,km/h$ of a car decreases the minimum time between two updates to 5 hours. For an inner city scenario with a maximum distance of 10 km and an MO walking with at most $v_{max} = 6\,km/h$, the minimum delay between two updates would be 1.66 hours.

These examples show that there are scenarios where the minimum update time would be hours rather than days. This would be sufficient for many "check-in" applications, as our evaluations based on real-world traces show in Section 8. For applications with shorter update intervals, the geometric approach would be better suited.

6. *PShare-GLM$_{map}$*: Geometric Position Sharing Using Map Knowledge

Until now, we assumed a free-space mobility model for *PShare-GLM* where MOs can be located at each position within $p(\pi, l)$. However, MOs positions are in many scenarios restricted to streets, places, buildings, etc. Therefore, an attacker could use map knowledge to decrease the effective area size of the obfuscation area $p(\pi, l)$ as shown in Fig. 5. The *effective area* of a MO's position $p(\pi, l)$ is defined as the area within $p(\pi, l)$ where the MO can actually be located.

To overcome this problem, we present *PShare-GLM$_{map}$* taking map knowledge into account. The general idea of our extension to *PShare-GLM* is to adapt MO positions in a first step to map knowledge such that each position $p(\pi, l'_i)$

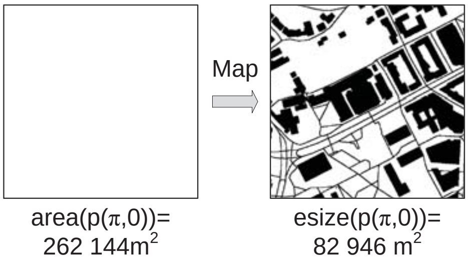


Figure 5: Map knowledge example reducing the effective area size ($b = 2, l_{max} = 9$)

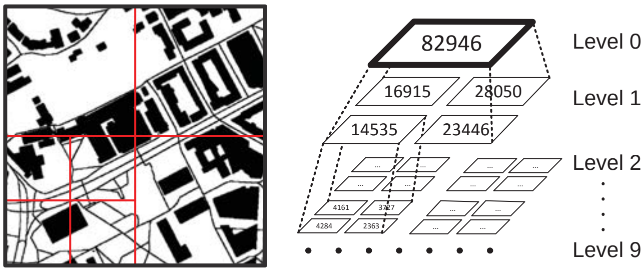


Figure 6: Quadtree and effective area example ($b = 2, l_{max} = 9$)

covers at least an area with an effective area size of a MO-defined value A_i. In a second step, we use the calculated positions for the share generation and apply the multi-secret sharing scheme as presented for *PShare-GLM*.

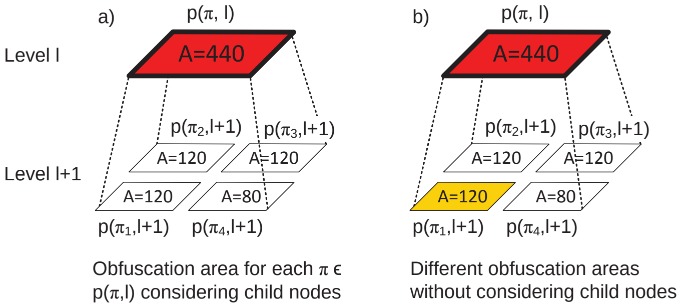
For the first step, the MO specifies its privacy requirements by defining different area values $A_i \in A_0, \ldots, A_m$ each representing the required minimum effective area size of precision level l'_i. The goal is now to find for each value A_i the smallest position $p(\pi, l'_i)$ such that the effective area size of $p(\pi, l'_i)$ is equal to or greater than A_i. To calculate the effective area size of position $p(\pi, l)$ we use function $esize(p(\pi, l))$. By using this notion, our goal is to find the minimum level l'_i such that $esize(p(\pi, l'_i)) \geq A_i$ for each $A_i \in A_0, \ldots, A_m$. For the sake of simplicity we consider from now on parameter $b = 2$ defining the granularity of the area of the MO's position. Nevertheless, our approach can also be applied to other parameter values of b.

We analyze map information in a preprocessing step to adapt positions to map knowledge. Thus, we calculate for each position whether or not it is a possible MO position, for instance, located on a street or in a building. To allow for an efficient search of $p(\pi, l'_i)$ for each A_i, we store all possible MO positions in a quadtree aggregating the effective area sizes for each level in a bottom-up approach (cf. Fig. 6). The quadtree has a depth of l_{max} and its root is defined by position $p(\pi, 0)$.

To calculate for a given position π and a required effective area size A_i the minimum level l'_i fulfilling $esize(p(\pi, l'_i)) \geq A_i$ without raising privacy concerns, we traverse the corresponding quadtree of π top down by evaluating the effective area size of position $p(\pi, l)$ for each level l and the four child nodes of $p(\pi, l)$ on level $l + 1$. If the described nodes of the quadtree cover an effective area of size equal to or greater than A_i, the quadtree is further traversed on the next level $l + 1$. Otherwise, level $l'_i = l$ is found.

To explain why it is required to consider $esize(p(\pi, l))$ for level l and also for the four child nodes of $p(\pi, l)$ on level $l + 1$ within each step when traversing the quadtree, consider the following example as depicted in Fig. 7:

Assume position $p(\pi, l)$ of an arbitrary level l covers the areas $p(\pi_1, l + 1)$, $p(\pi_2, l + 1)$, and $p(\pi_3, l + 1)$, all having an effective area size of 120. Area $p(\pi_4, l + 1)$ covered by $p(\pi, l)$ has an effective area size of 80. For a given threshold $A_i = 100$ the dark red area in Fig. 7a of level l would be calculated as obfuscation area for each position $\pi \in p(\pi, l)$ if the child nodes of $p(\pi, l)$ are considered as previously described. Without taking the child nodes of $p(\pi, l)$ into account, an attacker could increase his precision for an update in an area with an effective area size smaller than A_i based on the returned obfuscation area and the knowledge of the share generation algorithm. For instance, each



Obfuscation area for each π ϵ p(π,l) considering child nodes

Different obfuscation areas without considering child nodes

Figure 7: Example showing impact of considering child nodes (A_i=100, b=2)

position $\pi \in p(\pi_1, l+1)$ would lead to an obfuscation area of level $l+1$ marked in light yellow in Fig. 7b, whereas $\pi \in p(\pi_4, l+1)$ would lead to an obfuscation area of level l marked dark red. Thus, an attacker could increase his precision based on the returned obfuscation area by excluding $p(\pi_1, l+1)$ from a calculated obfuscation area of $p(\pi, l)$ for $A_i = 100$.

In the second step we calculate the secrets $K_1, \ldots, K_m$ that are used as input for the multi-secret scheme based on the calculated positions $p(\pi, l'_i)$ for each $A_i \in A_0, \ldots, A_m$. The number m of generated secrets is defined by the $m+1$ MO-defined area threshold values. Each secret K_i is calculated as the interleaved digits refining position $p(\pi, l'_0)$ of A_0 to position $p(\pi, l'_i)$ of A_i for $1 \leq i \leq m$.

The share generation of *PShare-GLM*$_{map}$ extends the share generation algorithm presented in Alg.1 for *PShare-GLM*. We apply the area adaptation to calculate the secrets by changing line 1 to 4 of Alg. 1 and therefore the way how the secrets are generated. Furthermore, we change the position provided within the m-share from $p(\pi, 0)$ to $p(\pi, l'_0)$.

Details of the share generation of *PShare-GLM*$_{map}$ are presented in Alg. 6. First, we calculate for each MO-defined threshold value $A_i \in A_0, \ldots, A_m$ the obfuscation area $p(\pi, l'_i)$ by using function $getMinLevelConsideringChilds(\pi, A_i, q)$ traversing quadtree q top-down by checking the previously described requirements for the child nodes and $esize(p(\pi, l)) \geq A_i$ for each position $p(\pi, l)$. Then, the obfuscation areas are used to generate the secrets K_i by interleaving the refining digits of position $p(\pi, l'_0)$ of A_0 to position $p(\pi, l'_i)$ of A_i. The calculated secrets are then used as input for the multi-secret sharing scheme as presented in Alg. 1 for *PShare-GLM*. Finally, position $p(\pi, l'_0)$ is stored within the m-share fulfilling $esize(p(\pi, l'_0)) \geq A_0$.

The share combination of *PShare-GLM*$_{map}$ is basically calculated as presented in Alg. 3 for *PShare-GLM*. The only difference is that position $p(\pi, l)$ in line 3 is now defined as $p(\pi, l'_i)$ and calculated by splitting up K_i into the x and y part refining $p(\pi, l'_0)$ of the m-share to $p(\pi, l'_i)$ by substituting the corresponding digits of $p(\pi, l'_0)$.

7. Security Analysis

In this section, we present the security analysis for *PShare-GLM* and *PShare-SLM*. We start with a description of the attacker model and an overview of the analyzed attacks, which are then discussed in detail.

7.1. Attacker Model

As attackers we consider malicious LSs and malicious clients. Each attacker has access to the public m-shares. Each malicious LS additionally knows one server r-share for each position. Each malicious client with access to a position of precision level l additionally knows l client r-shares (cf. Section 4). We both consider single attackers (a single malicious LS or client), as well as colluding attackers (multiple malicious LSs or clients). We structure the following analysis according to different attacks. First, we consider *single attackers* who analyze a single (current) position, or even the complete history of positions. Second, we analyze the effect of *colluding attackers* who put their compromised shares together. Since *PShare-GLM* and *PShare-SLM* are based on the same multi-secret sharing scheme, we do not distinguish between them unless the difference is relevant.

Algorithm 6 *PShare-GLM$_{map}$*: Map-based Share Generation (MO)

```
Function: $generate(\pi, l_{max}, n, \{A_0, \dots A_m\})$
q ← getQuadtree(π, l_max)
$q \leftarrow getQuadtree(\pi, l_{max})$
for $i = 0$ to $m$ do
    $p(\pi, l'_i) \leftarrow getMinLevelConsideringChilds(\pi, A_i, q)$
end for
for $i = 1$ to $m$ do
    $K_i \leftarrow interleave(refinement(p(\pi, l'_0), p(\pi, l'_i)))$
end for
$P, f(X) \leftarrow calculatePolynomial(K)$
$X \leftarrow n$ distinct integer
for all $x_i \in X$ do
    $y'_i \leftarrow f(x_i)$
    $sr^i_\pi \leftarrow (x_i, y'_i)$
end for
$m_\pi \leftarrow P, p(\pi, l'_0)$
return $m_\pi, \{sr^1_\pi, \dots, sr^n_\pi\}$
```

7.2. Single Attacker

First, we consider a malicious client having access to l client r-shares of a single position that can be used to reconstruct $p(\pi, l)$. Thus, the client knows secret K_l refining $p(\pi, 0)$ to $p(\pi, l)$ from these shares. As shown by Chan et al. [4], their multi-secret sharing scheme ensures that different secrets are independently protected by different polynomials. Thus, the information from K_l cannot be used to reconstruct other secrets and positions of levels greater l.

A single malicious LS has access to the m-share and one server r-share, i.e., it knows one distinct point of the secret polynomial $f(X)$. Therefore, the malicious LS can calculate for each precision level l with $0 < l \leq l_{max}$ exactly one point of the polynomial $f_l(X)$. Thus, the malicious LS can reconstruct the MO's position $p(\pi, 1)$ of level 1, while the positions of all levels greater 1, which require at least 2 r-shares, cannot be reconstructed.

Our proposed extension against map matching attacks (cf. Section 6) ensures that an attacker cannot reduce the effective area size of a known position $p(\pi, l'_i)$ below a MO-defined value A_i by using additional map knowledge.

Next, we consider further attackers knowing for a certain level l the complete position history $U = \{(p(\pi_{first}, l), t_{first}), \dots, (p(\pi_{last}, l), t_{last})\}$. Our algorithms create position shares of different positions independently from each other. Therefore, shares generated for $(p(\pi_i, l), t_i)$ cannot be combined with shares for $(p(\pi_{i+1}, l), t_{i+1})$. However, the reconstructed positions $p(\pi_{first}, l), \dots, p(\pi_{last}, l)$ could be used for a maximum velocity attack (cf. Section 4.5). Since we use delayed updates as counter measure, these attacks are also futile.

7.3. Colluding Attackers

Assume, for instance, three malicious clients c_A, c_B, and c_C. Assume that c_A and c_B own l client r-shares of the same precision level l so that both can calculate $p(\pi, l)$. c_C owns $l + 1$ client r-shares of the next precision level $l + 1$ to reconstruct $p(\pi, l + 1)$. Then, the collusion of c_A and c_B does not reveal anything new to c_A and c_B, as they were both already allowed to reconstruct $p(\pi, l)$ by calculating polynomial $f_l(X)$ using their l client r-shares each representing a distinct point of $f_l(X)$. Because polynomial $f_l(X)$ is uniquely defined by l distinct points, even using more than l client r-shares of the same precision level leads to the same polynomial $f_l(X)$ and therefore $p(\pi, l)$. Furthermore, the client r-shares of precision level l reconstructing $f_l(X)$ reveal nothing about the polynomial $f_{l+1}(X)$ and therefore about $p(\pi, l + 1)$. This is based on the fact that the polynomials of different precision levels are generated independently from each other in the multi-secret sharing scheme [4]. Thus, even the collusion of c_A and c_C with client r-shares of different precision levels does not reveal any new information. c_C can reconstruct $p(\pi, l + 1)$ even without collusion and the additional client r-shares of c_A carry no information about $f_{l+2}(X)$ of the next precision level $l + 2$.

Second, we consider multiple malicious LSs. Let m be the number of colluding LSs. These LSs can use their stored server r-shares to calculate m different client r-shares for each level l. Since we defined the threshold values as

Category	Pedestrian	Ground vehicle	Plane
Distance (d) and average speed (v)	$d \leq 10\,\mathrm{km}$ and $v \leq 6\,\frac{\mathrm{km}}{\mathrm{h}}$	($d \leq 10\,\mathrm{km}$ and $6\,\frac{\mathrm{km}}{\mathrm{h}} < v \leq 200\,\frac{\mathrm{km}}{\mathrm{h}}$) or ($10\,\mathrm{km} < d \leq 10\,000\,\mathrm{km}$ and $v \leq 200\,\frac{\mathrm{km}}{\mathrm{h}}$)	$d > 10\,000\,\mathrm{km}$ or $v > 200\,\frac{\mathrm{km}}{\mathrm{h}}$
$\mathbf{v_{max}}$	$6\,\frac{\mathrm{km}}{\mathrm{h}}$	$200\,\frac{\mathrm{km}}{\mathrm{h}}$	$1\,000\,\frac{\mathrm{km}}{\mathrm{h}}$
#updates	15 895 691	6 079 316	412 915

Table 1: Position update classification

$t_l = l$, l client r-shares are required to get a position $p(\pi, l)$ of precision level l. Therefore, m colluding LSs can reveal positions up to level $l = m$. This shows the desired graceful degradation of privacy property of our approach. The revealed precision increases with the number of compromised LSs. We could even increase the robustness by setting the threshold values t_l to values greater than l. Then more LSs are required to calculate a position of a certain level, which increases the overhead on the one hand, but also increases the robustness of our approach on the other hand. Therefore, our scheme allows for trading off overhead against robustness.

The collusion of malicious LSs and malicious clients is a special case of the collusion of malicious LSs. In this case, either the client with the highest precision level or the number of colluding LSs defines the revealed precision level.

8. Real World Trace Evaluation

As defined in Equ. 1, the minimum time between two updates is restricted by the MO's maximum speed and size of the level 0 position to guarantee protection against maximum velocity attacks. To analyze the practical impact of this restriction, we analyzed real datasets of position check-ins from existing location-based applications to see how they comply with this restriction. If many updates were violating the restriction, this would be an indication that our approach is not applicable to these applications since the user could not perform many desired updates.

The analyzed dataset, which was collected by [16] between September 2010 and January 2011, contains 22 387 922 user position check-ins of 224 803 users from different location-based applications all over the world. Since this dataset only contains geometric coordinates, we focus this evaluation on *PShare-GLM*. For our purpose, we processed the dataset as follows. We classified each position update based on the traveled distance and the average speed between two succeeding updates as shown in Table 1.

This table also shows the assumed maximum speed for *PShare-GLM* and the resulting number of updates per category. For each category, we calculated the percentage of updates that can be performed without violating the restriction. Figure 8 depicts the results for the categories and different sizes of level 0 positions ranging from 1 m (2^0) square side length to 32 768 m (2^{15}). As we can see, in the worst case for an obfuscation area side length of 32 768 m, 55.19% of the pedestrian updates are possible. For a side length of 1 024 m, which provides sufficient privacy for pedestrians in an innercity scenario for example, 88.09% of the updates are possible. For ground vehicles, 83.59% of the updates are possible using even the coarsest obfuscation of 32 768 m. Figure 9 depicts the results of all traces together from all categories. As it can be seen, for a level 0 size of 1 024 m, 90.08% of all updates can be published. Thus, we can state that the minimum update time restriction only affects a small number of updates.

9. Runtime Performance Evaluation

Besides security, the efficiency of share generation is important for the practical application of our approaches. Share generation is performed on the MO's mobile device, which typically has low performance in terms of computational speed. Also on such resource-poor devices, share generation must be possible in short time. An efficient share generation also leads to small overhead in terms of energy, which is desirable for battery-operated mobile devices.

We measured the overall time for share generation of *PShare-GLM*, *PShare-GLM$_{map}$*, and *PShare-SLM* on a state of the art mobile device (HTC Desire HD). For *PShare-SLM*, we used a hierarchy with a maximum level of $l_{max} = 3$ and measured the time to generate one m-share and three up to fifteen r-shares. The shares are all generated for the same number of $l_{max} = 3$ secrets such that three r-shares are required to reconstruct the precise position of the MO on

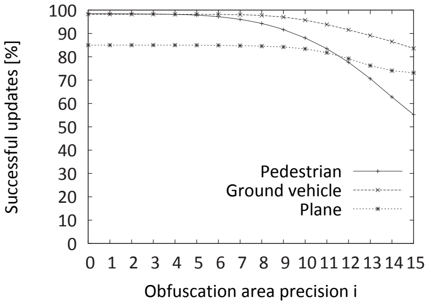


Figure 8: Successful updates for obfuscation areas of precision 2^i[m]

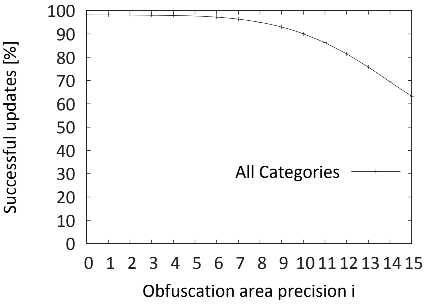


Figure 9: Successful updates for obfuscation areas of precision 2^i[m]

level three. For *PShare-GLM*, we measured the time to create one *m*-share and a varying number of $n = l_{max}$ *r*-shares. By increasing *n*, we also increased the number of used secrets. To evaluate *PShare-GLM*$_{map}$ we used road network and building data provided by OpenStreetMap [17] for the inner city of Stuttgart. We defined the obfuscation area of $p(\pi, 0)$ by setting $l_{max} = 13$, such that $p(\pi, 0)$ covers an area of 67.1 km^2 and has an effective area size of 10.5 km^2. We increased the number of generated *r*-shares from one to fifteen by increasing the number of used area thresholds. We varied each area threshold between 10 km^2 and 10 m^2. The results are shown in Fig. 10. The depicted results of *PShare-GLM* are measured for $b = 2$ only since other *b* values led to almost identical results. The plotted values are the average over several runs per share number using Google's micro-benchmarking tool Caliper.

As it can be seen, the runtime of *PShare-SLM* stays nearly the same for the different number of generated *r*-shares. This is based on the fact that the number of used secrets is defined by the maximum level three of the hierarchy. For *PShare-GLM* and *PShare-GLM*$_{map}$ the number of secrets was increased by increasing the number of generated shares. Nevertheless, the runtime of all three approaches stays well below 1 s even for larger share numbers. Therefore, we can state that the share generation algorithm is efficient and suitable even for resource-poor devices.

Next, we show the performance evaluation for the share combination, which is done by clients (location-based applications) typically running in an infrastructure without energy restrictions and high computational power. We measured for *PShare-GLM*, *PShare-GLM*$_{map}$, and *PShare-SLM* the time to combine the *m*-share with up to 15 *r*-shares on a state of the art personal computer (Intel Core 2 Duo, 2.53 GHz, 3 GB RAM). The results are shown in Fig. 11. As it can be seen, the time for share combination of all three approaches is nearly the same and stays below 1.1 milliseconds even for a larger number of combined shares.

10. Summary

In this paper, we presented a novel position sharing approach to manage private position information in non-trusted systems of third-party location services and clients. The basic idea is to split up the precise user position into position shares of limited precision, which are distributed to multiple location servers of different providers. Therefore, a single

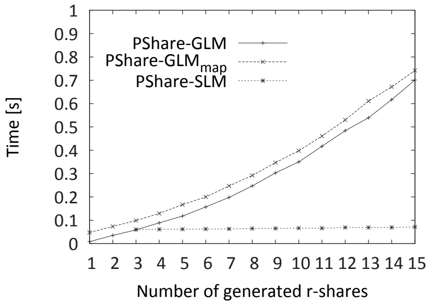


Figure 10: Share generation for $n = l_{max}$ shares, b=2

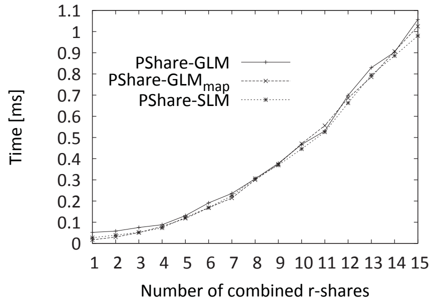


Figure 11: Share combination

compromised provider does only reveal a position of strictly limited precision. Clients are granted access to multiple shares from different servers, which can be combined to a position of higher precision to satisfy the individual quality requirements of different clients.

Our approach makes use of the concept of multi-secret sharing to calculate position shares. We have shown how to generate shares for geometric as well as symbolic positions, which demonstrates the versatility of the approach. Moreover, we included defense mechanisms against maximum velocity attacks, which are a serious threat to obfuscation-based mechanisms. Furthermore, we presented a map-based extension to our geometric position sharing approach that resists advanced attackers using map knowledge to increase precision. Finally, we showed the robustness, applicability, and runtime performance of the approach.

In future work, we will consider further semantic knowledge, for instance, periodic behavior of users, knowledge about points of interest, etc. that could be used by an attacker to increase precision.

References

[1] F. Dürr, P. Skvortsov, K. Rothermel, Position sharing for location privacy in non-trusted systems, in: Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications (PerCom 2011).

[2] P. Skvortsov, F. Dürr, K. Rothermel, Map-aware position sharing for location privacy in non-trusted systems, in: Proceedings of the 10th International Conference on Pervasive Computing (Pervasive 2012).

[3] DATALOSSDB, http://www.datalossdb.org/, 2012.

[4] C.-W. Chan, C.-C. Chang, A scheme for threshold multi-secret sharing, Applied Mathematics and Computation 166 (2005).

[5] P. Kalnis, G. Ghinita, K. Mouratidis, D. Papadias, Preventing location-based identity inference in anonymous spatial queries, IEEE Transactions on Knowledge and Data Engineering 19 (2007).

[6] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, L-diversity: Privacy beyond k-anonymity, ACM Transactions on Knowledge Discovery from Data 1 (2007).

[7] N. Li, T. Li, S. Venkatasubramanian, t-closeness: Privacy beyond k-anonymity and l-diversity, in: Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007).

[8] M. Duckham, L. Kulik, A formal model of obfuscation and negotiation for location privacy, in: Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005).

[9] G. Ghinita, M. L. Damiani, C. Silvestri, E. Bertino, Preventing velocity-based linkage attacks in location-aware applications, in: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS 2009).

[10] M. Damiani, C. Silvestri, E. Bertino, Fine-grained cloaking of sensitive positions in location-sharing applications, Pervasive Computing 10 (2011).

[11] H. Kido, Y. Yanagisawa, T. Satoh, An anonymous communication technique using dummies for location-based services, in: Proceedings of the International Conference on Pervasive Services (ICPS 2005).

[12] P. Shankar, V. Ganapathy, L. Iftode, Privately querying location-based services with sybilquery, in: Proceedings of the 11th International Conference on Ubiquitous computing (Ubicomp 2009).

[13] G. F. Marias, C. Delakouridis, L. Kazatzopoulos, P. Georgiadis, Location privacy through secret sharing techniques, in: Proceedings of the 1st International IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing (WOWMOM 2005).

[14] P. Gilbert, L. P. Cox, J. Jung, D. Wetherall, Toward trustworthy mobile sensing, in: Proceedings of the 11th Workshop on Mobile Computing Systems & Applications (HotMobile 2010).

[15] A. Shamir, How to share a secret, Communications of the ACM 22 (1979).

[16] Z. Cheng, J. Caverlee, K. Lee, D. Z. Sui, Exploring millions of footprints in location sharing services, in: Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM 2011).

[17] OpenStreetMap, www.openstreetmap.org, 2012.