# Optimized Location Update Protocols for Secure and Efficient Position Sharing

Zohaib Riaz, Frank Dürr, Kurt Rothermel
Institute of Parallel and Distributed Systems
University of Stuttgart, Germany
Email: {*zohaib.riaz, frank.duerr, kurt.rothermel*}@ipvs.uni-stuttgart.de

***Abstract*—Although location-based applications have seen fast growth in the last decade due to pervasive adoption of GPS enabled mobile devices, their use raises privacy concerns. To mitigate these concerns, a number of approaches have been proposed in literature, many of which rely on a trusted party to regulate user privacy. However, trusted parties are known to be prone to data breaches [1]. Consequently, a novel solution, called Position Sharing, was proposed in [2] to secure location privacy in fully non-trusted systems. In Position Sharing, obfuscated position shares of the actual user location are distributed among several location servers, each from a different provider, such that there is no single point of failure if the servers get breached. While Position Sharing can exhibit useful properties such as graceful degradation of privacy, it incurs significant communication overhead as position shares are sent to several location servers instead of one.**

**To this end, we propose a set of location update protocols to minimize the communication overhead of Position Sharing while maintaining the privacy guarantees that it originally provided. As we consider the scenario of frequent location updates, i.e., movement trajectories, our protocols additionally add protection against an attack based on spatio-temporal correlation in published locations. By evaluating on a set of real-world GPS traces, we show that our protocols can reduce the communication overhead by 75% while significantly improving the security guarantees of the original Position Sharing algorithm.**

***Keywords*-location-based services; privacy; efficient communication; dead reckoning; selective update**

## I. Introduction

With increased location-awareness of mobile devices, such as GPS/WiFi enabled smart phones and car navigation systems, location-based applications have found wide adoption in the past decade giving rise to novel concepts such as geo-social networking, live traffic updates, and pay-as-you-go car insurance. In spite of these beneficial advancements, sharing location data with non-trusted third parties raises privacy concerns. These concerns range from profiling of user behavior by advertizement agencies to personal security threats such as stalking [3].

In order to mitigate these privacy concerns, the research community has proposed a number of works as surveyed in [4]. Most of these works rely on a *trusted location service* which stores the location updates sent by the users. This service then allows the location-based applications to query user location in a privacy preserving manner. However, evidence, from incidents of data breaches at trusted entities [1] involving lost, stolen, and hacked data etc., suggests that the assumption of a trusted location service is at least questionable.

978-1-4799-5804-7/15/$31.00 

To this end, we introduced a novel approach called Position Sharing (PS) in [2] for secure sharing of location data with non-trusted parties. This scheme obfuscates the actual user location by splitting it into a number of imprecise pieces called *position shares*. For each location update, these position shares are generated and distributed to a corresponding set of location servers (LSs), each maintained by a different service provider. In this way, a non-trusted LS, when compromised, can only reveal limited information about the actual user location, thus exhibiting *graceful degradation* of user privacy. A location-based application, on the other hand, can query the LSs to acquire the shares and "fuse" them together to reconstruct the position of the user. The precision of the reconstructed position is determined by the number of fused shares. Therefore, by individually controlling the number of LSs a location-based application is authorized to access, a user can effectively *control the precision* of location data shared with them.

While PS avoids single point of failure with respect to user privacy, it incurs significant communication overhead in distributing position shares to multiple LSs. If user location is distributed among $n$ LSs, then each location update costs $n$ times the communication cost of updating a single LS when PS is not used. The problem is compounded when location updates are frequent, e.g. while sharing movement trajectories.

In this paper, we investigate the possibility of minimizing the communication overhead of PS when it is used to protect the location of a traveling user. We face two challenges while meeting this goal. First, the privacy guarantees provided by the PS algorithm need to be preserved. This is a requirement because our location update protocols alter the way in which the position shares are generated, thus possibly affecting the privacy guarantees. Second, the privacy guarantees should be maintained against attacks which exploit spatio-temporal correlation between location updates. This is a direct result of publishing frequent location updates as opposed to the single uncorrelated updates considered in the design of PS scheme.

Including and extending our initial attempt to solve the above stated challenges [5], we make, overall, three contributions. First, we integrate three location update protocols into PS for optimizing its communication overhead, namely, Dead Reckoning, Selective Update, and Selective Dead Reckoning. Second, we add protection to our protocols against a powerful correlation attack based on the knowledge of user's maximum traveling speed. This attack could otherwise significantly weaken the protection provided by PS. Finally, we analyze the communication efficiency and privacy security provided by our protocols on real-world GPS traces. Our evaluations show that most of our protocols reduce the communication overhead, at

To appear in the Proceedings of the 2nd IEEE International Conference on International Conference on Networked Systems (NetSys 2015), Cottbus, Germany, March 2015

least, by 75% while improving the security guarantees of the original PS algorithm.

The rest of the paper is organized as follows. In Section II, we briefly state the related work. Section III describes our system model and defines the problem statement. In Sections IV and V, we introduce the preliminary concepts, and the definitions of our update protocols, respectively. We then elaborate the addition of protection against correlation attack to our protocols in Section VI. Our evaluations are presented in Section VII. Finally, Section VIII concludes our paper.

## II. Related Work

With the gradual realization of location privacy concerns in the use of location-based applications, the research community has come up with a number of proposals in the last decade. Initially, the focus was on unlinking the identity of a user from the query they send to a location-based application. This protection goal gave rise to the root concept of location $k$-anonymity [6]. Here, user's location is represented by a region which contains at least $k$ other users such that the location-based application cannot identify the original generator of the received anonymous query. However, $k$-anonymity requires a trusted party which is knowledgeable about the location of all users in the system to generate the $k$-anonymous region.

To avoid trusted parties, a parallel research direction explores the idea of location obfuscation [7], [8]. With this method, the user's location is represented as a spatial region irrespective of the number of users it contains, so that the precise position of user inside the spatial region is protected from the location-based application. While location obfuscation can successfully avoid trusted mediating parties, it does not enable user-control over the *precision* of location information shared with varyingly trusted applications. In other words, the kind of protection is *all or nothing*, without any intermediate option for the user. PS, as proposed in [2], solves this problem while maintaining the notion of non-trusted system. It uses the concept of position shares which can be fused by location-based applications to get a position of defined precision.

As mentioned earlier, the location precision control provided by PS comes at the cost of additional communication overhead due to its distributed system architecture. As a solution to this problem, we seek help from the idea of dead-reckoning [9]. Dead reckoning introduces dynamic attributes, such as velocity, as part of a location update sent to the LS by the mobile object. Hence, the LS can predict the location of the mobile object in the absence of location updates while affording additional computational cost and a pre-defined, bounded error in the predicted location. When the error bound is exceeded, the mobile object updates the LS again with a fresh value of the dynamic attribute and its current actual location. We will use this idea in two of our optimization protocols.

With our aim to protect the privacy of a traveling user, attacks on location privacy based on spatio-temporal correlation between subsequent location updates cannot be ignored. One such basic but strong attack was introduced in [8] that relied on the simple knowledge of the maximum possible speed of the moving object. Using this knowledge, the adversary removed those parts of the published obfuscation regions which were unreachable, given the time and location of the last update, thus reducing user privacy. Against this attack, we adapt one of the solutions given in [8], called *delaying*, as an add-on to our proposed location update protocols.

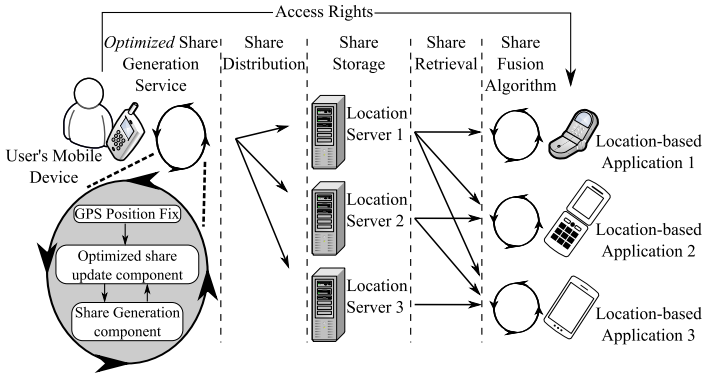



Fig. 1. System Model for *Optimized* Position Sharing.

## III. System Model and Problem Statement

In this section, we introduce our system model and define the problem statement.

### A. System Model

Our system, as illustrated in Fig. 1, consists of three types of components, namely, the mobile device, the location servers and the location-based applications.

The **mobile device** can calculate its current location $\pi$ using a built-in GPS receiver. It runs the *Optimized share generation service*, which represents $\pi$ as a set of location shares. These location shares are of two types: a single master share $s_{master}$, and $n$ refinement share vectors $S = \{\vec{s}_1, ..., \vec{s}_n\}$. $s_{master}$ represents the most obfuscated form of $\pi$ and, therefore, is published to all of the $n$ location servers. On the other hand, the refinement shares $\vec{s}_i \in S$ which decrease the imprecision of $s_{master}$, are distributed, one to each location server (LS).

The **LSs** manage the location shares sent by the mobile device and implement access control mechanisms to enforce their authorized retrieval by the **location-based applications**. These applications acquire access rights to a number of LSs directly from the user. They then retrieve the location shares and fuse them together to reconstruct a position of higher precision than $s_{master}$ using the *share fusion algorithm*.

In contrast to the simple share generator in [2], our optimized share generation service, as shown in Fig. 1, adds an additional *optimized share update component* that implements our protocols for efficient location updates. This component exclusively accesses the position fixes generated by the GPS receiver, and based on the decision from the underlying location update protocol, invokes the *share generation component* to generate new location shares when necessary.

Whenever the share generation component is invoked, it generates $\{s_{master}, S\}$ in such a fashion that on fusion, they allow reconstruction of the actual location $\pi$ at various precision levels. The precision of the reconstructed position increases with the number of fused shares. To better understand this concept, consider the example illustrated in Fig. 2. Part (a) of the figure represents $s_{master}$ as circle $c_0$ with center $p_0$ and radius $r_0 = \phi_{min}$, where $\phi_{min}$ is the minimum precision set by the user. As for the refinement shares, they are vectors that translate the center of a circle to a new position. Whenever the center is translated, the radius is decreased by $\Delta_\phi = \phi_{min}/n$,

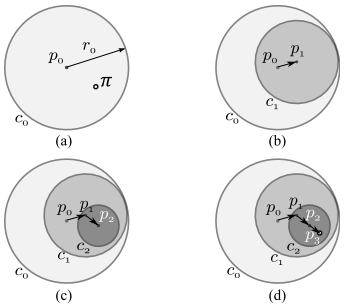



Fig. 2. Geometric representation of the actual location $\pi$ in PS. (a) $s_{master}$ with $\pi$ shown inside it. (b)-(d) $n = 3$ shares incrementally fused to gradually increase precision of user location until $p_3 = \pi$ is calculated in (d).

to increase the precision of the fused position. In parts (b)-(d) of Fig. 2, refinement shares are gradually added, or fused, to $p_0$ to achieve positions of higher precision, i.e., $p_1$ to $p_3$. Note that, per addition of a vector $\vec{s}_i$, there is a constant decrease of $\Delta_\phi$ in radius of the obfuscation circle. Note also that any position $p_i$, represented by the circle $c_i$, results from the addition of $i$ refinement vectors on top of $s_{master}$ and that $p_3 = p_n = \pi$ since we use 3 refinement shares in this example.

For the rest of the paper, we will use $p_n$ and $\pi$ interchangeably to refer to the actual user location.

### B. Problem Statement

The design of our location update protocols aims to minimize the communication overhead of the original PS algorithm while preventing any compromise on the location privacy guarantees it provides.

More formally, a location update in the *non-optimized* PS case implies sending a refinement share $\vec{s}_i$ and the master share $s_{master} = \{p_0\}$ to each of the $n$ LSs. With $\vec{s}_i$ and $p_0$ being 2-dimensional variables, we quantify the cost of updating location over a trajectory of $m$ position fixes as the number of variables sent to the $n$ LSs, i.e.:

$$\text{Cost}_m^{PS} = m \cdot \underbrace{\{ \overbrace{n \cdot 2}^{\vec{s}_i} + \overbrace{n \cdot 2}^{s_{master}} \}}_{\text{single location update}} \quad (1)$$

With integration of our location update protocols, the *optimized* PS minimizes $\text{Cost}_m^{PS}$ by modifying the location update process so as to skip the location updates, either fully or partially. Therefore, the individual counts of the sent $\vec{s}_i$ and $s_{master}$ determine the total cost for the $m$ position fixes:

$$\text{Cost}_m^{PS_{opt}} = num(s_i) \cdot 2 + num(s_{master}) \cdot size(s_{master}) \quad (2)$$

Here, the size of $s_{master}$ varies to include the user's velocity when Dead-Reckoning is used.

The modification of the location update process by our protocols leads to two problems. First, it introduces error in the user location present at the LSs. Against this, we require that the introduced error $\epsilon$ should not exceed a user-defined bound of $\epsilon_{max}$ at any instant during the course of the trajectory:

$$\epsilon < \epsilon_{max} \quad (3)$$

Second, a privacy adversary could exploit their knowledge of the share generation process and of our update protocols to determine a non-uniform distribution of actual user location $\pi$ inside the obfuscation region, thus reducing privacy. The requirement, therefore, is that this non-uniformity of distribution, as quantified later in Section VII-B, for optimized PS should be less than that for the non-optimized PS.

Finally, we also assume that the privacy adversary has knowledge of the user's maximum possible travel speed, i.e., $v_{max}$. They exploit $v_{max}$ to determine and remove those parts of the current obfuscation region which cannot be reached from the last obfuscation region considering time between the two updates, thus decreasing the intended privacy. To avoid this attack, our protocols must update locations in such a manner that any position inside the published obfuscation areas is reachable by the user. For the rest of the paper, we will refer to this as the *maximum movement boundary* (MMB) attack.

## IV. Preliminaries

In order to better understand the design of our location update protocols as detailed in Section V, we will first briefly look at the share generation process of the original PS algorithm from [2] and its corresponding privacy adversary model.

### A. Share Generation in PS

During the share generation process, each share is generated such that the following conditions are met. First, each circle $c_i$ must contain the actual location $p_n$. Second, each $c_i$ is fully contained in $c_{i-1}$. Consequentially, a third requirement is that the length of any share vector should not exceed $\Delta_\phi$. While meeting these restrictions, two algorithms, the *a-posteriori* and *a-priori*, were proposed which fundamentally differed in the order of generation of $p_0$.

The *a-posteriori* algorithm generates the set $S$ of refinement share vectors first and calculates $p_0$ at the end by subtracting the sum $\sum_{\forall \vec{s}_i \in S} \vec{s}_i$ from $p_n$. Share vectors are generated randomly, with lengths in the interval $[0, \Delta_\phi]$ and no restrictions on direction. Therefore, all shares are uncorrelated to each other. A notable property of this algorithm is that the uncorrelated shares cause $p_n$ to be *normally distributed* around $p_0$ within the circle $c_0$. As more shares are fused to $p_0$, the distribution of $p_n$ around the fused position, say $p_k$ inside the circle $c_k$ converges to a uniform one.

On the other hand, the *a-priori* algorithm first generates a uniformly random location for $p_0$ in a circle of radius $\phi_{min}$ around $p_n$, and then finds the set $S$ of share vectors that, on fusion, lead from $p_0$ to $p_n$. However, because the overall direction and distance to be covered by the refinement shares in $S$ is already defined by the vector $\overrightarrow{p_0 p_n}$, the resulting shares are correlated, i.e., the definition of each share is dependent on the definition of shares generated before it.

For the rest of the paper, we will refer to the above two share generation algorithms as SGAs.

### B. Adversary Model

The *privacy adversary* considered in [2] has knowledge of the SGAs as well as access to $k$ out of the $n$ total refinement shares. The adversary exploits this knowledge to generate the $(n - k)$ refinement shares that are unknown to them and use these to derive a better estimate $\pi_{attack}$ of user location inside the obfuscation region $c_k$. Therefore, from the adversary's point of view, the actual user location $p_n$ has a non-uniform distribution inside the obfuscation region $c_k$ instead

of a uniform one that was intended for maximum privacy. The success of this attack is quantified by the distribution $P_{attack}(\phi)$ which measures the probability that the estimated position $\pi_{attack}$ lies closer to the actual location $p_n$ than a distance of $\phi$ units. Note that it is assumed that the user can move freely in space without any restrictions, and that, the adversary does not have access to map knowledge.

## V. Update Protocols

We next explain three different location update protocols, namely, Dead Reckoning, Selective Update, and Selective Dead Reckoning, which reduce the communication overhead of the original PS approach.

### A. Dead Reckoning (DR)

With its existing use in efficiently updating locations of moving objects [9], DR is a natural choice for our investigation. The basic idea behind DR is to enable the LS to predict the location of the moving object, even in the absence of location updates. To achieve this, the LS models the movement of the moving object as a prediction function $f_{pred}$ such that location at a future time $t$, say $loc^{'}(t)$, can be estimated given the knowledge of the last known location $loc(t_{last})$ and linear velocity $\vec{v}_{t_{last}}$ (see Eq. (4)). For each position fix generated at the moving object, it also executes $f_{pred}$ and then determines the deviation of location $loc^{'}(t)$ from its actual location $loc(t)$, e.g. the euclidean distance. If this deviation exceeds by a certain predefined threshold $th_{DR}$, the moving object sends a fresh location update to the LS, thus resetting the incremental error integrated in the estimated location $loc^{'}(t)$.

$$loc^{'}(t) = f_{pred}(loc(t_{last}), \vec{v}_{t_{last}}, t) \quad (4)$$

The basic idea of applying DR to position sharing is as follows. After a $s_{master}$ and the set of $n$ refinement shares $S$ are generated and distributed to the LSs, we apply DR to the position of the $s_{master}$ (center of $c_0$, i.e., $p_0$). The set of $n$ refinement shares $S$ remain unchanged. If the error $\epsilon$ in $p_0$, exceeds the threshold $th_{DR}$, we calculate a new $s_{master}$ together with $S$ and update the LSs. In order to limit the error $\epsilon$ below the user-defined maximum error value of $\epsilon_{max}$, we set $th_{DR} = \epsilon_{max}$.

Figure. 3 shows the details of our algorithm. At the start of the main loop, the algorithm waits until a current user location $p_n(t)$ is fixed by the mobile device (line 3). If there was already a last update sent to the LSs, i.e., $upd_{last} = \{s_{master}, S\}$, then the algorithm uses it to calculate the LS's prediction, i.e., $p_0^{'}$ at the current time $t$ (lines 5-6). Note that in the case of DR, $s_{master}(t) = \{p_0(t), \vec{v}_t\}$. Then, last update's shares $upd_{last}.S$ are fused with the calculated $p_0^{'}$ to get estimated $p_n^{'}$ in line line 7. In lines 8-9, new shares are set to be generated if the error $\epsilon$ in estimated $p_n^{'}$ is greater than the threshold $th_{DR}$. If shares are to be generated anew, either because they do not exist (for the first location update), or $th_{DR}$ was violated, the algorithm generates and distributes them in lines 12-17. Fresh calculation of the velocity $\vec{v}_t$ is done in line 13. Finally, the generated shares along with the freshly calculated $\vec{v}_t$ are distributed to the $n$ LSs in line 15.

```
1:  procedure DEAD RECKONING
2:      loop over the whole trajectory
3:          p_n(t) = wait for next position fix
4:          new_shares = FALSE
5:          if exists(upd_last) then
6:              p_0'(t) = f_pred(upd_last.s_master, t)
7:              p_n'(t) = fuse_shares(p_0'(t), upd_last.S)
8:              if ε = dist(p_n(t), p_n'(t)) > th_DR then
9:                  new_shares = TRUE
10:             end if
11:         end if
12:         if !exists(upd_last) OR new_shares then
13:             v_t = calc_attributes()
14:             {p_0(t), S(t)} = gen_shares(p_n(t), n, φ_min)
15:             Distribute{s_master(t), S(t)} → n LSs
16:             upd_last = {s_master(t), S(t)}
17:         end if
18:     end loop
19: end procedure
```

Fig. 3. Algorithm for Dead Reckoning with PS

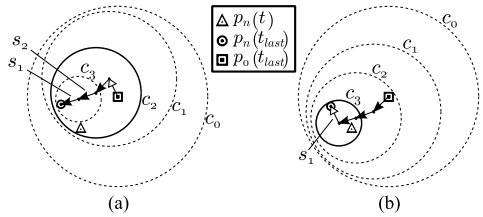



Fig. 4. Importance of share ordering in SU. (a) New position fix $p_n(t)$ falls in $c_2$ requiring update of $s_1$ and $s_2$. (b) Re-ordered shares bring $p_n(t)$ inside $c_3$ requiring update of $s_1$ only.

### B. Selective Update

Selective Update (SU) is based on the idea of partial re-usage of refinement shares. On generation of a new position fix, only a minimum number of refinement shares are re-generated and updated at their respective LSs, such that the user location at the LSs is consistent with the actual one. Therefore, SU requires 1 to $n$ update messages per location update instead of the fixed $n$ messages for non-optimized PS, thus reducing communication overhead.

Formally, SU is only successful if the new position fix at time $t$, i.e., $p_n(t)$, lies within the circle $c_0$ around the last generated $p_0$, i.e., $p_0(t_{last})$. In other words, the distance of $p_n(t)$ from $p_0(t_{last})$ is less than the threshold $th_{SU}$, which can at most be equal to the minimum precision $\phi_{min}$ (see Eq. (5)). Note that, in contrast to DR, the user location with the LSs in the case of SU has no error.

$$dist(p_0(t_{last}), p_n(t)) < th_{SU}, 0 < th_{SU} \leq \phi_{min} \quad (5)$$

For SU, it is important to determine an ordering for shares that decides which shares should be re-generated in order to have minimal updates. As an example, Fig. 4(a) shows a set of already existing shares for a given last position fix $p_n(t_{last})$. As the new position fix $p_n(t)$ (the triangle) arrives, at least two shares ($s_1$ and $s_2$) must be updated to represent it. This is because $p_n(t)$ does not lie inside the circle $c_3$ but rather in $c_2$. If however, we change the fusion order of shares (switch the black and white headed arrows as shown in Fig. 4(b)), $p_n(t)$ can be brought inside $c_3$, requiring only one share update ($s_1$).

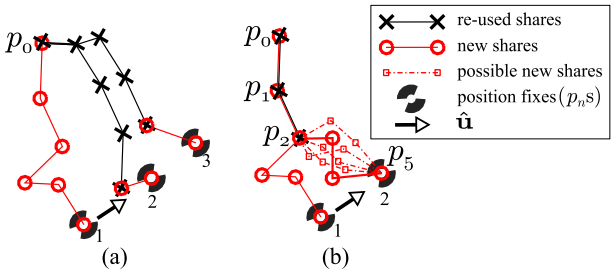



Fig. 5. Selective Update. (a) Optimized version $SU_{com}$ with one new share for fixes 2 and 3. (b) Non-optimized version $SU_{sec}$ with many possible share sets.

Intuitively speaking, as movement is the cause of inconsistency of shares, an optimal share ordering should change those refinement shares which have the smallest components along the direction of movement, i.e., $\hat{\mathbf{u}} = \frac{\overrightarrow{p_n(t_{last})p_n(t)}}{|\overrightarrow{p_n(t_{last})p_n(t)}|}$. This optimization is easily realized by sorting the shares in ascending order of their components along $\hat{\mathbf{u}}$, i.e., $\vec{s}_i.\hat{\mathbf{u}}$, and updating the shares with the least components first. However, this optimization can adversely effect the privacy security of the PS algorithm.

For an explanation, consider a second example shown in Fig. 5(a) which implements this optimization. Here shares are generated anew for position fix 1 whereas SU is applied to represent position fixes 2 and 3. Note that SU first sorts the shares as discussed. As a result, it requires only one share update per new position fix. However, as the single share update must account for the displacement between the consecutive position fixes, the overall set of shares get more and more aligned along the vector $\overrightarrow{p_0p_n}$ over the course of a few position fixes. This alignment increases the *correlation* between the shares which can lead to weakening of privacy security of the PS algorithm as will be demonstrated by our evaluations in Section VII. We call this optimized but vulnerable version of SU as $SU_{com}$.

In contrast, Fig. 5(b) represents a non-optimized version of SU where an update of three shares is required to represent position fix 2. However, more share updates also imply more flexibility in their definition. Referring again to Fig. 5(b), we see that the three shares translate the position $p_2$ to the actual position fix $p_5$. As the length $|\overrightarrow{p_2p_5}|$ in this case is small (compared to the maximum aggregated possible length of the 3 shares, i.e., $3.\Delta_\phi$), many sets of 3 shares are possible between $p_2$ and $p_5$ (dotted lines in the figure). Our algorithm generates many such sets and finally chooses that set for selective update which maximizes the deviation of its constituent vectors from the average share vector, i.e., $\frac{\overrightarrow{p_0p_5}}{n}$. Hence, the updated shares have less chances of alignment along $\overrightarrow{p_0p_n}$, and consequently, less correlation among them. We will refer to this more secure, non-optimized, version of SU as $SU_{sec}$.

Figure 6 gives the pseudo-code of the function that performs SU for both $SU_{com}$ and $SU_{sec}$. Given the last sent location update $upd_{last} = \{s_{master}, S\}$, current position fix $p_n$, unit vector $\hat{\mathbf{u}}$ representing movement direction, and the threshold $th_{SU}$, the function first tackles, in lines 2-3, the case where SU is not applicable with reference to Eq. (5). Then, in case of the optimized version of selective update, i.e, $SU_{com}$, all shares $s_i \in upd_{last}.S$ are sorted in line 6. Next, the function determines, in line 8 the number $k$ of shares to be re-generated, by selecting the smallest obfuscation circle $c_{n-k}$ inside which

```
1:  function SELECTIVE UPDATE(upd_last, p_n, û, th_SU)
2:      if dist(upd_last.s_master.p_0, p_n(t)) ≥ th_SU) then
3:          return null
4:      else
5:          if comm_optimization_enabled then
6:              upd_last.S = sort_shares(upd_last.S, û)
7:          end if
8:          k = num_of_shares_to_update(upd_last, p_n)
9:          new_shares = gen_k_shares(upd_last.S, p_n, k)
10:         return updated_shares
11:     end if
12: end function
```

Fig. 6. The Selective Update function

$p_n$ resides. Finally, the $k$ new shares are generated in line 9. If $k > 1$, our algorithm selects the best, out of a number of possible sets of $k$ shares as illustrated in Fig. 5(b), so as to avoid the introduction of correlation between shares.

### C. Selective Dead Reckoning

We observed from our initial experiments that DR and SU had interesting individual properties. While DR results in very less location updates on straight, predictable parts of the user trajectory, it performed poorly at jerky patches such as sharp turns. On the other hand, SU, while not as efficient as DR, was relatively stable against sharp turns. Therefore, we define Selective Dead Reckoning (SDR) which uses SU on top of DR in order to achieve a more stable overall performance.

In SDR, whenever $th_{DR} = \epsilon_{max}$ is violated, we need to choose between generating a complete set of new shares, as in the original DR algorithm (see Fig. 3), or performing a selective update as in the SU function (see Fig. 6). To make this decision, we define a heuristic based on the minimal cost of one location update in DR and SU. As the LS additionally needs the velocity $\vec{v}_t$ in SDR to perform DR, thus $s_{master} = \{p_0, \vec{v}_t\}$, and the cost of a share generation is $\text{Cost}_1^{SDR} = \text{Cost}_1^{DR} = (\text{Cost}_1^{PS} + 2 \cdot n)$. Lets assume that DR in SDR fails at the $q^{th}$ position fix after the last share generation. Note here that the cost of DR over the $q-1$ position fixes was zero but it would rise to $\text{Cost}_1^{DR}$ if DR is continued on the $q^{th}$ position fix. Instead, if SU was performed over the $q$ fixes, it would cost at least a size of $(q \cdot 2)$, with a minimal one share update per position fix. Comparing the two costs, we define our heuristic condition for activating SU as $\text{Cost}_1^{DR} > (q \cdot 2)$, which, considering Eq. (1), is reduced to:

$$3n > q \tag{6}$$

The pseudo-code for SDR is almost the same as that of DR, apart from an additional call to the SU function (between the lines 8-10 of the DR algorithm in Fig. 3), based on the satisfaction of above inequality.

## VI. Protection against the MMB Attack

Even with the optimized location updates, our protocols do not ensure that the user can reach all parts of the published obfuscation regions from their last location, considering their maximum speed. Therefore, the adversary could cut out the unreachable areas from the published regions, thus realizing the MMB attack. This threat occurs at the time of new share

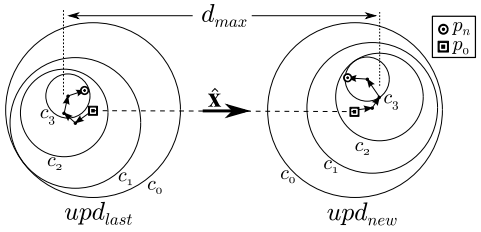



Fig. 7. Determination of $d_{max}$ in order to counter the MMB attack.

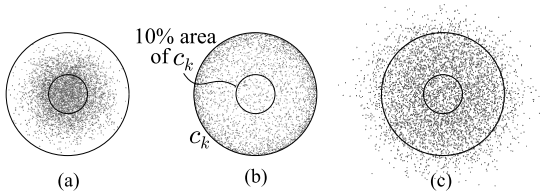



Fig. 8. Distributions of $p_n$ around $\pi_{attack}$. (a) High density. (b) Low Density. (c) The seemingly low density case of $P_{SA}$

generations when there is a major shift of position of $p_n$ inside $c_0$, due to regeneration of $p_0$. Note that, for PS, we must consider MMB attacks at all levels of precision, i.e., $p_0$ to $p_n$. Therefore, whenever a complete set of new shares is generated i.e. $upd_{new} = \{p_0, S\}$, we determine the maximum distance $d_{max}$, which must be reachable since the last update $upd_{last}$ so that all obfuscation regions of $upd_{new}$ are valid i.e. the user could be located anywhere inside them. This maximum distance is measured between the centers of corresponding obfuscation regions, $c_0$ to $c_n$, of $upd_{new}$ and $upd_{last}$.

For example, consider the two updates shown in Fig. 7. Here the maximum distance $d_{max}$ turns out to be between $c_2$ of both updates. Note that a different share fusion order may result in another pair of $c_i$ as having a different $d_{max}$. To get the maximum possible value of $d_{max}$, we calculate the unit vector $\hat{\mathbf{x}}$ for the vector $\overrightarrow{upd_{last}.p_0 upd_{new}.p_0}$. Then we sort the shares of $upd_{new}$ and $upd_{last}$ in descending order of their components along and opposite to $\hat{\mathbf{x}}$ respectively. Finally, we fuse the shares in this order and determine $d_{max}$, which represents the distance that must be traversable within the time between the two updates, i.e., $\Delta t = (t_{new} - t_{last})$, when the user travels at their maximum speed $v_{max}$. This forms the condition, as defined below, which when satisfied, enables MMB safe publishing of the new location update.

$$\frac{d_{max}}{v_{max}} < \Delta t \tag{7}$$

If, however, the user requires more time than $\Delta t$ to traverse $d_{max}$ at $v_{max}$, then we skip the publishing of the calculated shares $upd_{new}.S$ until an upcoming position fix satisfies Eq. (7) for the same set of shares. Note that $upd_{new}.p_0$ is re-calculated as per the upcoming position fix by subtracting from it the sum of the shares $upd_{new}.S$. As the MMB attack comes into play only at new share generations, and as we do not discard any set of generated shares for protecting against it, our protection algorithm is not expected to change the privacy guarantees of the original *a-posteriori* and *a-priori* SGAs.

## VII. Evaluations

Finally, we evaluate the efficiency, in terms of communication overhead of our optimized location update protocols as well as their privacy properties.

### A. Performance Metrics

For a given trajectory of $m$ position fixes, we measure the reduction in the size of overall communication achieved by our protocols compared to the original PS algorithm. To this end, we define the Reduction Ratio (RR) metric as:

$$RR = (1 - \frac{\text{Cost}_m^{PS_{opt}}}{\text{Cost}_m^{PS}}) * 100 \tag{8}$$

where $\text{Cost}_m^{PS_{opt}}$ and $\text{Cost}_m^{PS}$ are defined in Section III-B.

We define a second performance metric with regards to the MMB attack. While our protection against it does not affect the share generation process, it does involve inhibiting location updates for position fixes where Eq. (7) is not satisfied. We quantify this effect by measuring the proportion of position fixes for which location updates can be safely published. We call this metric the publishing ratio (PR). PR is defined for a trajectory with $m$ position fixes as:

$$PR = \frac{num(MMB\_safe\_updates)}{m} \tag{9}$$

### B. Privacy Metrics

As seen in Section V, the partial updates in SU and SDR protocols effect the share generation process in a non-deterministic fashion, due to the non-deterministic user movement. Therefore, the adversary cannot generate unknown shares by the sole knowledge of our algorithm, which disables the basic attacker model which allows the determination of $P_{attack}(\phi)$, as discussed in Section IV-B. However, we identify two fundamental features that the adversary can still exploit to determine a distribution similar to $P_{attack}$. With $k$ shares already known, these features are the distribution of calculated position $p_k$ inside the obfuscation region $c_k$, and the correlation among the $k$ shares. We now define corresponding to these features, two probabilistic metrics of attack to quantify the privacy guarantees of our optimized PS approaches.

For the first feature, i.e., distribution of $p_k$ inside $c_k$, we define the Distribution attack probability $P_{dist}(\phi)$. This probability simply treats $p_k$ as the adversary's estimate of user location $\pi_{attack}$ and determines its proportion that falls within a distance of $\phi$ units from $p_n$. $P_{dist}(\phi)$ is intended to capture the convergence of $p_n$ to a normal distribution around $p_0$ in the case of *a-posteriori* SGA.

For the correlation feature, we introduce the Share Average (SA) attack probability $P_{SA}(\phi)$. This probability, unlike $P_{dist}(\phi)$, is estimated by generating the unknown shares as the average of the $k$ known shares. $\pi_{attack}$ is then determined by fusing the known and generated shares. By averaging the known shares, $P_{SA}(\phi)$ captures the pre-existing correlation among them, as in the *a-priori* SGA, as well as the correlation introduced by re-usage of shares in SU, and SDR.

With $P_{dist}(\phi)$ and $P_{SA}(\phi)$ being independent measures of $P_{attack}(\phi)$, the adversary is interested in the more non-uniform out of these two probability distributions in order to better determine the actual location of user inside the obfuscation region. To this end, they estimate $P_{dist}$ and $P_{SA}$ when $\phi$ represents the radius of $10\%$ area of the obfuscation region $c_k$. If $P \in \{P_{dist}(\phi), P_{SA}(\phi)\}$ deviates from the ideal case of $P = 0.1$, i.e. a uniform distribution, this leads to two cases of high and low density as shown in Fig. 8(a) and (b) respectively.

In the case of $P_{SA}$, however, low density values, i.e. less than $0.1$, do not confirm high density of $p_n$ in the rest of the $90\%$ area of $c_k$. This is because, for $P_{SA}$, $p_n$ could lie outside the boundaries of $c_k$ as shown in Fig. 8(c). Therefore, $P_{SA} < 0.1$, is not considered useful by the adversary.

For $P \in \{P_{dist}(\phi), P_{SA}(\phi) > 0.1\}$, the adversary determines its non-uniformity as follows. If, for instance, $P = 0.7$, then it represents a dense distribution and, compared with the highest density case of $P = 1$, has a non-uniformity value of $w(P) = (0.7-0.1)/(1-0.1) \sim 0.67$. In contrast, if $P = 0.03$, it represents low density distribution and, compared with the lowest density case of $P = 0$, has a non-uniformity value of $w(P) = (0.1 - 0.03)/(0.1) = 0.7$. Therefore, with a higher value of $w(P)$, $P(\phi) = 0.03$ is more non-uniform compared to $P(\phi) = 0.7$ in this case.

The above discussion is summarized by a third privacy metric $P_{best}(\phi)$, which selects the more non-uniform, out of $P_{dist}(\phi)$ and $P_{SA}(\phi)$, as the adversary's best estimate:

$$P_{best} = \begin{cases} P_{SA} & \text{if } w(P_{SA}(\phi) > 0.1) > w(P_{dist}(\phi)), \\ P_{dist} & \text{otherwise} \end{cases} \quad (10)$$

### C. Evaluation Setup

To evaluate our protocols, we downloaded real-world GPS traces from [10] and categorized them as *long-route, urban travel, unstructured-walk and structured-walk*. This categorization is important to judge the viability of the protocols in real life usage. The categories offer different average distance $\Delta_{dist}$ and angle changes $\Delta_{angle}$ per position fix, as well as a range of average speeds (see Table I).

We developed and tested our protocols in Octave [11]. For our evaluations with the four trace categories, we have defined values of the various parameters as given in Table II. We average our results of each protocol by running it 10 times on all trace categories. Furthermore, while the SU protocol has the two types $SU_{com}$ and $SU_{sec}$ as discussed in Section V-B, the same branching also applies to SDR.

### D. Results for Communication-overhead

As mentioned earlier, we measure the reduction ratio (RR) of the size of total communication over a trajectory. Table III gives the RRs for each protocol, per trace category. We present the average results for both, the *a-posteriori* and *a-priori* SGAs, as they did not differ significantly. $DR$ with a minimum RR of $44.2\%$ shows the most variation in performance,

TABLE I. THE DATASET OF GPS TRACES AND ITS CHARACTERISTICS.

| Category | Num. of Traces | Avg. Speed (km/h) | Avg. $\Delta_{dist}$ (m) | Avg. $\Delta_{ang}$ (deg) | Std. $\Delta_{ang}$ (deg) |
|---|---|---|---|---|---|
| i. long-route (car) | 4 | 78.3 | 31.4 | 1.7 | 6.6 |
| ii. urban travel (car) | 4 | 24.8 | 55.8 | 12.4 | 21.5 |
| iii. unstructured-walk | 3 | 5.1 | 30.7 | 14.6 | 17.8 |
| iv. structured-walk | 7 | 6.4 | 24.7 | 22.5 | 29.8 |

TABLE II. PARAMETERS FOR EVALUATIONS

| Cat. | $n$ | $\phi_{min}$ (m) | $th_{DR}$= $\epsilon_{max}$ (m) | $th_{SU}$ (m) | $v_{max}$ (km/h) |
|---|---|---|---|---|---|
| i. | 5 | 3000 | 40 | 2500 | 150 |
| ii. | 5 | 1000 | 30 | 800 | 80 |
| iii. | 5 | 500 | 15 | 400 | 10 |
| iv. | 5 | 500 | 15 | 400 | 10 |

TABLE III. AVERAGE RR FOR *a-posteriori* AND *a-priori* SGAS (%)

| Cat. | $DR$ | $SU_{com}$ | $SU_{sec}$ | $SDR_{com}$ | $SDR_{sec}$ |
|---|---|---|---|---|---|
| i. | 91.0 | 88.1 | 84.6 | 95.1 | 94.8 |
| ii. | 44.2 | 82.3 | 76.0 | 80.8 | 78.8 |
| iii. | 52.8 | 81.7 | 75.3 | 85.8 | 84.3 |
| iv. | 57.6 | 83.6 | 79.2 | 86.8 | 85.5 |
| ***min.*** | **44.2** | **81.7** | **75.3** | **80.8** | **78.8** |

TABLE IV. PRS WITH MMB PROTECTION (%)

| ***a-posteriori* SGA** | | | |
|---|---|---|---|
| **Cat.** | $DR$ | $SU_{sec}$ | $SDR_{sec}$ |
| i. | 48.9 | 91.6 | 86.3 |
| ii. | 67.2 | 98.7 | 99.6 |
| iii. | 38.7 | 98.1 | 99.7 |
| iv. | 40.0 | 91.5 | 89.4 |
| ***min.*** | **38.7** | **91.5** | **86.3** |
| ***a-priori* SGA** | | | |
| i. | 35.5 | 58.5 | 64.2 |
| ii. | 56.0 | 81.8 | 90.9 |
| iii. | 28.5 | 63.9 | 77.8 |
| iv. | 30.2 | 65.9 | 69.7 |
| ***min.*** | **28.5** | **58.5** | **64.2** |

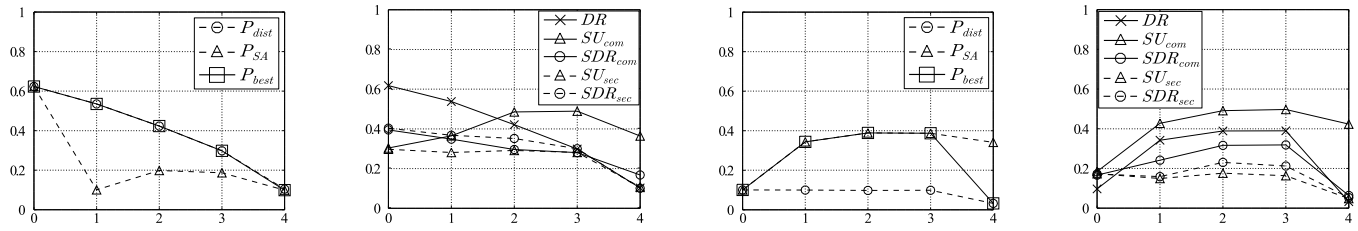
attributable to the varying predictability of trajectories in each trace category as captured by the standard deviation of $\Delta_{ang}$ in Table I. Both flavors of $SU$, i.e., $SU_{com}$ and $SU_{sec}$, have relatively invariant performance with minimum RRs of $81.7\%$ and $75.3\%$ respectively, whereas $SDR_{com}$ and $SDR_{sec}$ have close or higher RRs than $DR$, and the corresponding $SU$ protocols. It is noticeable that the secure versions of $SU$ and $SDR$ have at most $\sim 6\%$ and $\sim 2\%$ lower RRs, respectively, as compared to their optimized versions for the four trace categories. However, this reduction provides improved security guarantees as will be seen later. In general, all protocols, except for $DR$, have minimum RRs of above $75\%$.

### E. Effects of MMB protection

Next, we analyze the effects of adding protection against MMB attack to $DR$, $SU_{sec}$, and $SDR_{sec}$. Our test revealed that there is no notable change in the values of RRs by this addition. As for the publishing ratio (PR), Table IV summarizes results for both SGAs. In general, we observe that minimum PR of our protocols is significantly lower for *a-priori* SGA than in *a-posteriori*. This is explained by the fact that in *a-posteriori* SGA, $p_0$ is normally distributed around $p_n$ in contrast to its uniform distribution in *a-priori* SGA. With reference to Eq. (7), this leads to comparatively smaller values of $d_{max}$ for *a-posteriori* SGA which takes less time to traverse at $v_{max}$. Consequentially, Eq. (7) is satisfied for more location updates, making them safely publishable. From among our protocols, $DR$ performs poorly with low minimum PRs of $38.7\%$ and $28.5\%$ for the *a-posteriori* and *a-priori* SGAs respectively. However, $SU_{sec}$ and $SDR_{sec}$ perform significantly better with similar minimum PRs of close to $90\%$ and $60\%$ for the two SGAs. The low performance of $DR$ compared to the other two protocols is due to the overall poor predictability of the trajectories. This leads to more and more new share generations which in turn proportionally increase the number of occasions where updates must be inhibited due to the MMB attack.

### F. Results for Privacy Security

Finally, we evaluate our optimized protocols for PS and the original PS algorithm against the probabilistic metrics




(a) $P_{attack}$ vs. $k$ (*a-posteriori*). (b) $P_{best}(\phi)$ vs. $k$ (*a-posteriori*). (c) $P_{attack}$ vs. $k$ (*a-priori*). (d) $P_{best}(\phi)$ vs. $k$ (*a-priori*).

Fig. 9. Privacy Security against $k$ compromised shares. (a) and (c): results for non-optimized PS. (b) and (d): corresponding results for optimized PS.

of attack: $P_{dist}(\phi)$, $P_{SA}(\phi)$, and $P_{best}(\phi)$. Recall that given $k \in \{1..(n-1)\}$ shares are compromised by the adversary, $P_{dist}$ considers $p_k$ as $\pi_{attack}$ whereas $P_{SA}$ generates the $(n-k)$ unknown shares, by averaging the $k$ shares, and determines $\pi_{attack}$ by fusing them on top of $p_k$.

Figure 9(a) and (c) show the results of Monte Carlo simulation (1000 runs) for the original *a-posteriori* and *a-priori* SGAs when $\phi$ is set equal to the radius of $10\%$ area of the obfuscation region $c_k$. While ideally, i.e., for uniform distribution of $p_n$ inside $c_k$, the probabilities $P_{dist}(\phi)$, and $P_{SA}(\phi)$ should be $0.1$ for all values of $k$, this is not the case. As for *a-posteriori* SGA, see Fig. 9(a), it is weaker against $P_{dist}$ than $P_{SA}$ with a high probability, i.e, $62\%$, of finding $p_n$ in $10\%$ region around $p_0$ due to its normal distribution. On the other hand, for *a-priori* SGA, $P_{SA}$ evaluates to higher values with increasing $k$ due to the correlation between the generated shares (see Fig. 9(c)). Note that in both of the figures, $P_{best}$, represented by solid line, captures the more non-uniform of $P_{dist}$ and $P_{SA}$ (see Eq.(10)).

We now look at the average results of our protocols, as shown in Fig. 9(b) and (d), when applied to all trace categories using the *a-posteriori* and *a-priori* SGAs respectively. For ease of inspection, we only plot $P_{best}$. As $DR$ does not modify shares, it gives the expected result of maintaining the shape of the original $P_{best}$ curve for both SGAs. $SU_{com}$, on the other hand, results in more correlation between shares which is evident by the rise of its curve to reach $50\%$ in case of both SGAs resulting in poor security guarantees. One noticeable improvement from $SU_{com}$, however, is the reduction of $P_{best}$ from $62\%$ for original *posteriori* SGA to $30\%$ when $k = 0$. This is due to the gradual movement of $p_0$ away from $p_n$ as shares get stretched due to their re-use, thus causing the transformation of the normal distribution of $p_n$ to a more uniform one around $p_0$. $SU_{sec}$ with its relatively more share updates and flexibility in share definition, however, performs best for both SGAs by being closest to the probability of $0.1$ for all values of $k$. Finally, $SDR_{com}$ and $SDR_{sec}$ provide similarly improved security for the original *a-priori* SGA. For *a-posteriori* SGA, however, the improvement in security for $SDR_{sec}$ is considerably more as it is the second nearest, after $SU_{sec}$, to the probability of $0.1$ for all values of $k$.

In short, all our protocols, apart from $SU_{com}$ improve the security guarantees of the original SGAs. Our tests confirmed that these guarantees do not change by the addition of protection against the MMB attack.

Considering both, the communication efficiency and the privacy guarantees, we can conclude that the secure versions of $SU$ and $SDR$ provide our best solutions with consistent performance over all trace categories. Not only do they provide high values of above $75\%$ for the minimum RR, they also improve the privacy guarantees of the original PS algorithm.

## VIII. Conclusion and Future Work

In order to make the solution of Position Sharing feasible for preserving location privacy, we have presented a suite of location update protocols, based on the concepts of dead-reckoning and selective update, which significantly reduce its communication cost while improving its privacy security guarantees. We have discussed the detailed design and integration of these protocols into Position Sharing approach, followed by evaluations on real world traces as proof of their performance.

As future work, we will add further attack capabilities in our privacy adversary model. These will primarily include map-based knowledge such as the road-network topology.

## IX. Acknowledgment

This work is part of the PriLoc project (Privacy-aware Location Management) of the University of Stuttgart funded by the German Research Foundation (DFG) grant RO 1086/15-1.

## References

[1] (2014) World's biggest data breaches & hacks — information is beautiful. http://tinyurl.com/lgyx9lc.

[2] F. Dürr, P. Skvortsov, and K. Rothermel, "Position sharing for location privacy in non-trusted systems," in *Proc. of the IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, 2011, pp. 189–196.

[3] B. Lucas. (2014, 09) Stalkers turn to mobile technology and social media to harass victims — herald sun. http://tinyurl.com/mdmsyg4.

[4] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 163–175, 2014.

[5] Z. Riaz, "Optimized position update protocols for secure and efficient position sharing," Master's thesis, 2013.

[6] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. of the 1st Int. Conf. on Mobile systems, applications and services (MobiSys)*. NY, USA: ACM, 2003, pp. 31–42.

[7] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Pervasive Computing*, ser. LNCS. Springer Berlin Heidelberg, 2005, vol. 3468, pp. 152–170.

[8] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, "Preserving user location privacy in mobile data management infrastructures," in *Privacy Enhancing Technologies*, ser. LNCS. Springer Berlin Heidelberg, 2006, vol. 4258, pp. 393–412.

[9] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and querying moving objects," in *Proc. of the 13th Int. Conf. on Data Engineering*, 1997, pp. 422–432.

[10] Gpslib - gps tracks hosting service. [Online]. Available: http://gpslib.net/

[11] Gnu octave. [Online]. Available: http://www.gnu.org/software/octave/