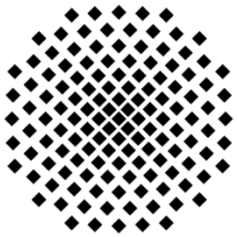


Institut für Parallele und Verteilte Systeme
Abteilung Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

IPVS

Master Thesis Nr. 3458

Optimized Position Update Protocols for Secure and Efficient Position Sharing

Zohaib Riaz

Studiengang:	INFOTECH
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	Dr. rer. nat. Frank Dürr
begonnen am:	29th October 2012
beendet am:	30th April 2013

Abstract

The large scale availability of sophisticated devices such as the mobile phones has opened a new paradigm of possibilities in terms of the services that can be provided to common masses. Most mobile phones nowadays come with integrated GPS receiver which can provide the user with their accurate location information. This has resulted in many common mobile applications which use this location data to provide services to users such as the point of interest finder, friend finder and navigation services etc. The location data from the GPS receiver is uploaded to Location Servers (LSs) by the user's device. Location Based Services, LBSs, running on client device communicate with remote LSs to download a user's location data and process it appropriately to offer client the desired service. If a malicious third party can get access to the location information on LSs, a user's privacy can significantly be violated. This can result in serious concerns being raised over the security of the user.

Many approaches have been proposed to sufficiently protect user privacy in location services. Most of them rely on the availability of a trusted third party (TTP) [1] at the LS's end. However, the assumption of a TTP is justifiably unrealistic [2] as events have been recorded where trusted parties have given up private user information to unauthorized clients. To overcome this problem, a novel approach was proposed in [3] which assumes partially trusted LBSs and LSs. This approach breaks down the position information into a number of shares and distributes each share to a separate LS managed by a different operator. The location information in any one share is equal and can be incrementally compiled by a LBS based on the number of shares it can access. The user grants access rights to different LBSs according to their trust on them. The overall effect is to have a possibility of graceful degradation of privacy if some of the location shares are compromised. There is however a price paid in the form of higher communication load. When the mobile object sends an update, it has to send a share to each LS. Similarly, when a client needs a particular user's location, it has to query a number of LSs according to its access rights. Consequently, a higher communication cost is incurred which is undesirable for implementation of these privacy schemes.

This thesis studies the possibility of applying traditional as well as subjective approaches for reduction of communication overhead in Position Sharing (PS) algorithm. From among traditional approaches, we have focused on Dead Reckoning (DR) where LSs predict the user location based on last location update and additional information about user movement. This results in less location updates with the compromise of incurring a known amount of bounded inaccuracy in the predicted position calculated by the LSs. A subjective approach of Selective Update is chosen for analysis according to the structure of PS algorithm. In this approach, reduction in communication is achieved by updating only the minimum necessary shares for short movements made by the user. From the observed behavior, a third technique as a merger of the mentioned techniques is also inferred. Finally, the viability of these approaches as possible solutions for reducing communication overhead in PS algorithm is studied and their results are evaluated on a set of real world GPS traces.

Acknowledgements

First of all, I am very thankful to Dr. Frank Dürr for his valuable time and suggestions during supervision of my thesis. I would also like to thank Prof. Kurt Rothermel for giving me an opportunity of working in his department.

Finally, I would like to recognize the continuous support of my family members which has always been there throughout my studies.

Table of Contents

List of Figures

List of Tables

Acronyms

GPS	Global Positioning System
LS	Location Server
LBS	Location Based Service
TTP	Trusted Third Party
PS	Position Sharing
DR	Dead Reckoning
QoS	Quality of Service
SU	Selective Update
SUaDR	Selective Update and Dead Reckoning
NN	Nearest Neighbor
kNN	*k* Nearest Neighbor
AS	Anonymizing Server
SA	Share Average
OSPS-ASO	Open Space Position Sharing with Any Share Order
CSPS-FSO	Constrained Space Position Sharing with Fixed Share Order
PMR	Percentage Message Reduction
MPPF	Messages Per Position Fix

1. Introduction

1.1 Mobile Technology and Location Data

In the last decade, mobile phones have found large acceptance in consumer market. The nature of mobile communication systems inherently requires coarse knowledge of the position of the user in order for providing the communication services. A cell phone reports its position to the cellular network periodically in order to be reachable by the network. There are already concerns raised about the amount of data collected and kept by the mobile phone companies about their subscribers [4]. The potential uses of this location data range from analysis of people's interests, designing of better marketing strategies etc. to helping in criminal activity tracking.

The sophisticated technology and the abundant computing power of smart-phones have re-defined user experience. The idea of installable software applications for mobile devices has caught a big share of attention from the software industry. For smart-phone applications, the knowledge of user location gives an important insight into user context. A main source of this location context is an integrated GPS receiver in smart phones. Although its accuracy is affected by the number of available satellites and signal strength, a sufficiently precise position can be estimated by study of consecutive position updates.

A range of applications have spawned which provide very intelligent services such as searching nearest restaurants, gas stations etc, based on user location. The precision of location data provided to an application directly affects its Quality of Service (QoS) [3]. For example, an application providing navigation service will require highly precise user location in order to correctly determine appropriate route to user destination, especially in urban scenarios with a dense network of roads. Similarly, consider an application that searches nearest gas station. In this case, the nearness of reported results, which determines the QoS, will depend upon the precision of the user location used by the LBS.

1.2 Location Based Services and their implications on User Privacy

Mobile applications also allow users to upload location data to Location Servers (LSs). These applications usually use Location Based Service (LBS) on the World Wide Web. Some commonly known examples of LBSs, in social networking, are Google's Latitude and Yahoo' Fire Eagle. These LBSs host the location updates from their users. When a client with sufficient access rights queries this location data, they are allowed to download the location information by the LSs.

There are a number of reasons for the existence of a third party Location Server (LS) over a direct broadcast or multicast of position updates. The server acts as a buffer to give users a freedom of sending an update at any time. Similarly, it also

allows a client to query location of a user at any particular time. This existence of a LS removes a lot of communication overhead and synchronization complexity that would arise if users are performing a broadcast or multicast to share their locations. On top of that, an LS, while centrally having the knowledge of locations of all users and additional landmarks can run server side software to process complex queries, such as range queries, nearest restaurant search etc., for which the user's device does not have enough information. Apart from a clear reduction in communication bandwidth, this also saves a lot of computation power which is an important factor considering most clients use battery powered mobile devices with limited computational resources. To keep unwanted clients from accessing user data, access control is implemented centrally by the LS [3]. Therefore, access to a user's location data is only allowed to those clients who have the requisite permissions of access granted by the user.

Social networking services allow users to set automatic location updates to keep their circle of friends informed over their location. However, location data can potentially reveal a lot of information about a person to an attacker. For example, an attacker might infer the location of a person's home and office by analyzing their location with respect to time of day [5]. Similarly, information like what time a person leaves their homes for office, how much time they spend there, when they are present at their homes etc can easily be termed as sensitive when it comes to privacy, and consequently, security.

Many approaches have been proposed to meaningfully quantify and increase user privacy in use of LBSs. A number of these approaches depend on a Trusted Third Party (TTP)[1] added on the server side. A TTP can intentionally add confusion in the user's identity, thereby protecting the privacy of the user [6]. However, it is of utmost importance that the location data stored at the LS is handled discretely and protected appropriately from possible snooping or stealing. As there are doubts about data security on the LS [7], it implies that the user should have some amount of control over how much they want to reveal about their location to the LS. These doubts are further corroborated by popular ideas in technologies which are based on mass information analysis such as user profile based advertizing and surveillance by law enforcement agencies etc.

1.3 Position Sharing: User's approach to Privacy Control

By assuming a partially trusted system, there is a requirement for initial processing of location information on user's mobile device before it can be uploaded to the LS or processed by a LBS. In this respect, a novel technique called, Position Sharing (PS) was presented in [3]. This technique breaks down the user position into parts which can be combined to reconstruct accurately the original position. The idea is to spread the pieces of position information among more than one LSs where each piece contributes equally in reducing the uncertainty when the original position is being reconstructed. In this way, a non-trusted LS can only deliver limited information to a malicious agent if it is compromised. Also, due to the breakdown of position information, different levels of privacy can be maintained by the user with LBSs of different trustworthiness. In case of a few shares are compromised by a LS or a LBS, the result is a graceful degradation of user's privacy as only a part of location information is given up.

The type of position obfuscation that results from PS has been theoretically analyzed as a possible solution to sufficiently securing privacy [3]. There is however a clear communication overhead generated for the implementation of this algorithm. Where in a non-privacy aware system we have a single message sent to the LS, now, by incorporating PS, we have a message for each LS for every position update. Therefore, if n shares are generated by the user's device, then the overhead due to PS multiplies total messages by n. Consequently, the high cost of communication poses a serious barrier in feasibility of Position Sharing as a practical approach for securing location privacy in mobile systems.

1.4 Overview and Contributions

Even in non-privacy aware systems, it is considered of utmost importance to reduce the communication overhead as much as possible to reduce overall communication cost [8]. This thesis will focus on achieving optimization of communication overhead for PS algorithm. In this regard, we will consider Dead Reckoning (DR), Selective Update (SU) and a merger of these two called Selective Update and Dead Reckoning (SUaDR).

DR is a famous approach for reducing the communication overhead in non-privacy aware location based systems [8, 9]. This implies that the LS is able to predict, with bounded error, the location of a user based on their last position update and additional data provided to help in position prediction. The result is a reduction in the number of required location updates to the LS with increase in computational cost and addition of imprecision in the calculation of user position.

The idea of SU dictates that all shares should not necessarily be generated anew for each location update; rather old shares should be reused as much as possible. The change in location from the last position fix to the new fix is represented by re-generation of minimum number of old shares such that they account for the change in position. Therefore, a considerable reduction in the number of messages to the LSs is achieved compared to a naïve implementation of PS.

We will also propose a third technique by merging the above two calling it SUaDR. It will be shown that this merger performs better then the individual DR and SU algorithms by exploiting good attributes of both with regards to communication overhead reduction and privacy protection. All evaluations are done on a set of real world GPS traces. It will be shown that an average reduction in messages of 75% is consistently achieved on different sets of GPS traces while considerably maintaining the privacy security provided by the basic PS algorithm.

In the Chapter 2, we will discuss the previous research work done in the area of privacy protection. The system model of PS with our optimizations and the problem statement will be elaborated in Chapters 3. In Chapter 4, we will explain Position Sharing algorithm. Our formalization of overhead optimization techniques will be done in Chapter 5. Finally, the implementation and evaluation of these techniques will be discussed in detail in Chapter6.

2. Background and Related Work

This chapter is intended to give a broad overview of the previous work done in the area of privacy protection as researched in [1]. First, we will give a brief overview of the aspects of location privacy that are meant to be protected. Then, some of the popular privacy securing approaches will be discussed. Finally, the common attacks on location privacy against which protection is aimed by privacy protection approaches will be elaborated.

2.1 Aims of Privacy protection [1]

The fact that privacy is a very abstract social concept requires the need of a clear definition. In context of location data, a person can be assumed to have control over their privacy if they can decide when and when not, do they want to reveal their identity, their location and the time of being at that particular location to another person. These three aspects of a person's location information, namely identity, spatial and temporal, and their combinations can result in different standards of privacy suited for a variety of situations. Each of these can significantly affect the amount of information kept from or conveyed to the attacker.

2.2 Privacy Protection Approaches involving TTPs

A TTP is trusted with accurate user location data and therefore, it is a potential single point of failure in the system. However, considering the complexity of the problem, many approaches for protecting user privacy have been developed with the assumption that a TTP does exist in the overall scenario. Some of these approaches will now be discussed.

2.2.1 *K*-Anonymity

A LBS running on a web server requires client location as part of the query sent to it. This is a pre-requisite for range queries or Nearest Neighbor (NN) search. However, revealing actual user position to the LBS is not desired as it is not trusted. So an Anonymizing Server, (AS), which is assumed to be a TTP, is introduced in the system [6] as shown in Figure 2.1.

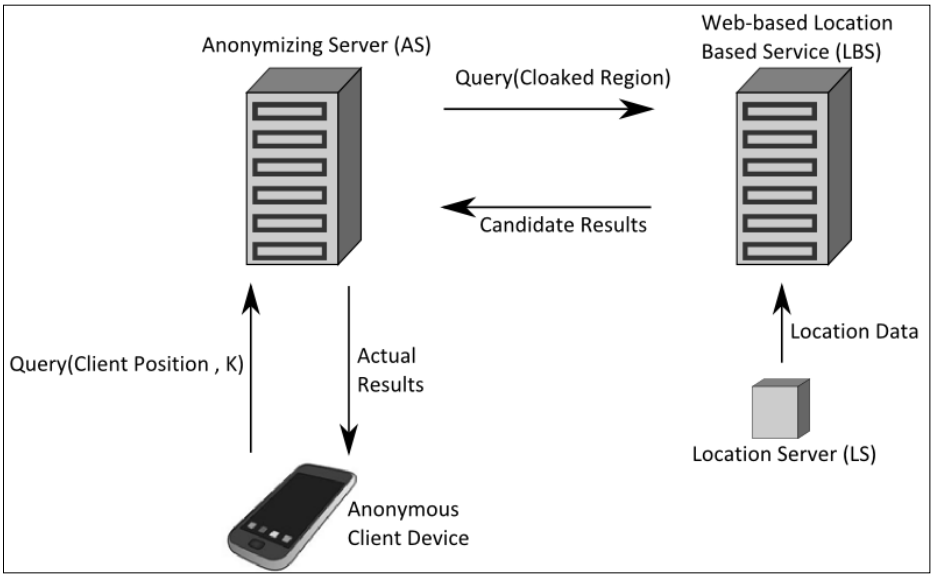


Figure 2.1. Privacy protecting LBS querying using Anonimizing Server [6].

The AS directly receives the request from the clients through a secure connection. It removes the identity of the user and sends the modified query to the LBS on behalf of the client. But because the query is with respect to the actual user location, the location is still revealed to the LBS. To avoid this, the K-Anonymity approach defines a cloaking region, instead of the user location, to query the LBS. This region includes the locations of at least *K-1* other users who are also querying the LBS. The LBS now processes the query for a region instead of a precise user location and hence more than desired results are produced and communicated to the AS. The AS then filters out the irrelevant results based on its knowledge of client's actual position and returns the accurate results to the client.

K-Anonymity has be used to achieve protect user's identity and their spatial and temporal information. A few of its extensions described in [1] are breifly discussed below.

2.2.1.1 l-diversity

To protect the user's semantic location e.g. a hospital, the extension of *l*-diversity was proposed in [10]. This approach ensures that the other *K-1* users are at a uniquely different and considerable distant semantic location before forming a *k*-cluster. In this way, sufficient confusion is produced for each semantic location of the user inorder to protect from an attacker who already has or attempts to gain knowledge of users contextual whereabouts. However, the *k*-cluster formed by this approach might be unrealistic. For example, in a locality where the distribution of one type of semantic location e.g. a bar, is high as compared to others, the attacker can ignore the possiblity of the user being at places of low distribution e.g. church, hospital, cinema etc. The assumption of the attacker in ignoring the low distribution locations as possible user locations is reasonable, because the distribution of locations in the *k*-cluster doesnot reflect the actual distribution of locations in the locality. Therefore presence of too many low distribution places in the *k*-cluster will not be affective for protecting user privacy.

2.2.1.2 t-closeness

As an extension on *l*-diversity, another approach by the name *t*-closeness was presented in [11], which improves the distribution of semantic locations in the *k*-cluster. This approach defines a measure to quantify the distance in distribution of a particular attribute in members of the *k*-cluster and the distribution of the same attribute in the overall population of users. This distance is kept under the desired value *t* for the sensitive attribute being protected. If *t* is small, then an attacker should not be able to converge to the actual user location by ignoring improbable locations in the *k*-cluster.

2.2.1.3 p-sensitivity

If any particular attribute of the members of the *k*-cluster does not have enough distinct values, then instead of increasing user privacy, this can result in helping an attacker in increasing their knowledge about the member under attack. For example, if each member of the *k*-cluster is moving at low average speed, then the attacker will be sure that his victim is also moving at low speed. To avoid this, the concept of *p*-sensitivity was used in [12]. *p*-sensitivity ensures that each attribute of the members of *k*-cluster have at least *p* distinct values.

2.2.2 Mix Zones

The concept of *Mix* Zones is explained in [13]. All user locations are sent to a middleware. The middleware divides the map into regions which are called zones. Any application that requires user location has to get it through the middleware. The user can permit applications to access their location in some zones, called *application* zones, and declare other zones as *Mix* zones. In *Mix* zones, the user's identity is confused with other users present in the zone. The middleware represents each user's identity as a pseudonym which changes when the user moves between an *application* and a *mix* zone. An attacker in the form of a malicious application can use their knowledge of user position in *application* zone to estimate their exit from the *mix* zone. But the mapping of user pseudonyms before entering a *mix* zone to their new pseudonyms after leaving the *mix* zone is computationally expensive. The tradeoff of privacy protection in *mix* zones comes as low QoS by applications because of poor position accuracy. Another disadvantage is the reliance on a TTP in the form of the middleware.

2.3 Privacy Protection Approaches without a TTP

Data mishandling incidents in the past [2] have led to development of mistrust on TTPs. Therefore, avoiding a TTP in a privacy protection scheme is considered a plus. Among the many approaches developed in this direction, we will consider a few popular ones for a brief discussion.

2.3.1 Location Obfuscation

This approach, used in [14], aims to protect the spatial aspect of user's privacy. The idea is to allow the user to control the accuracy of their published location. If the user desires a less accurate position to be published, the necessary inaccuracy is converted to a relative measure of position degradation. This

degradation is then realized by using a set of possible location obfuscation techniques; uncertainty radius increase, location shift and even a decrease in uncertainty radius as shown in Figure 2.2. Position degradation by increase in uncertainty radius is obvious. However, for the cases of position shift and uncertainty radius decrease, the degradation is possible due the decrease in the probability of finding the user in the new obfuscation area. Therefore, the possibility of the user being located outside the obfuscation region exists.

Although location obfuscation achieves the user-desired privacy requirement, the QoS provided by the LBS is directly affected. Also, the user-defined inaccuracy is present in all user locations sent to the server. Apart from deterioration of QoS from the LBS, location obfuscation is also inflexible due to its rigid definition of location privacy level.

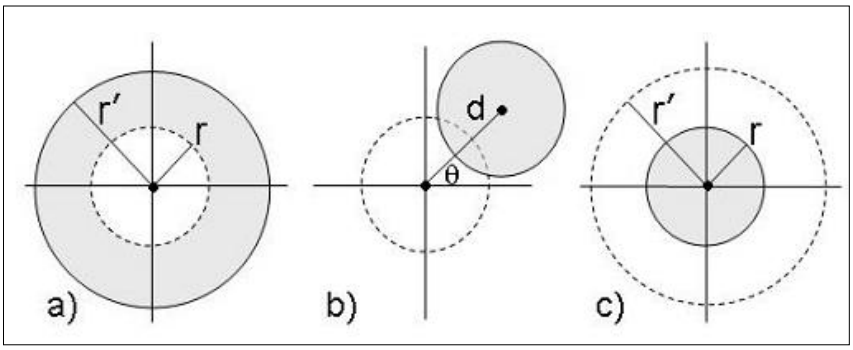


Figure 2.2. Means of acheiving Obfuscation in [14]. a) Enlarging uncertainty radius in original user position. b) Shifting user position. c) Decreasing uncertainty radius.

2.3.2 Dummy Positions

As the name suggests, this approach is based on generation of many false positions by the user for each actual position fix. These locations along with the actual user location are sent to the LBS for querying [15]. In this way, the user's original location is kept private as the LBS is not able to distinguish actual user location from dummy ones. The user's device can, however, filter out the exact results because it knows its position. This process is illustrated in Figure 2.3.

For preserving user identity in a sequence of set of generated positions, the dummy positions produced in each step appear in the neighborhood of the positions in the last step. However it is difficult to avoid learning of the distribution of dummy generation scheme over time by the attacker. To overcome this problem, a dummy generation scheme based on historic traffic data was presented in the *SybilQuery* approach [16]. The infeasibility of access to traffic history data, especially for mobile users, is a major obstacle in application of this approach.

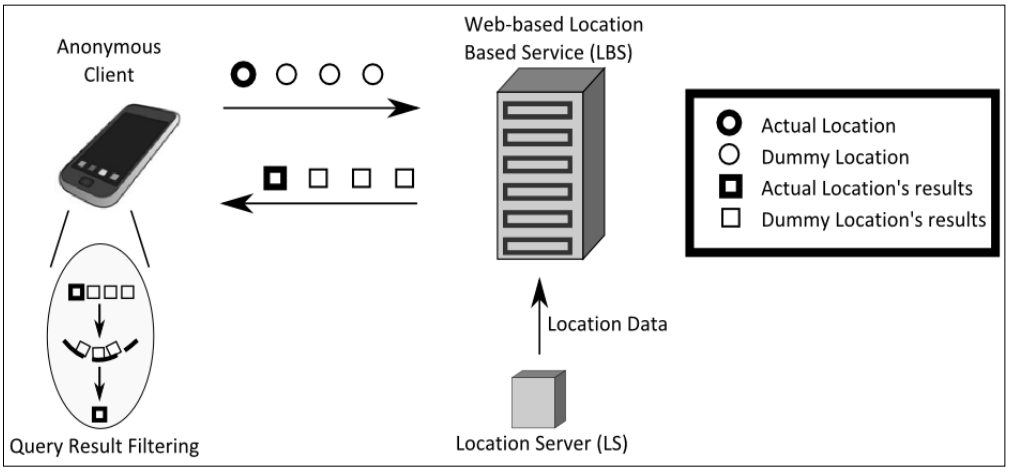


Figure 2.3. Use of Dummy Positions for securing user privacy.

Overall, the cost of communication is high with dummies approach. This is mainly due to sending of dummy positions by the user and their associated results returned by the LBS.

2.3.3 Coordinate Transformation [17]

This approach is very similar to position obfuscation although it maintains an overall high QoS at the expense of extra communication cost. The original user position is transformed using primitive operations such as rotations and shifts and sent to the LBS. If a client downloads this position, they must know the transformation that should be inversely applied on this data to achieve the original user position. So the user should relay the original transformation to the clients causing high communication overhead.

In a technique specialized for *k*-Nearest Neighbor (kNN) queries, called SpaceTwist [18], an *anchor* location is generated near the original user location. The server processes the query according to the *anchor* location, assuming it to be original user location, and returns results in ascending order of their distance from this location. The user can then process the returned results to rearrange them in ascending order from their actual location.

2.3.4 Encryption of Location Data

By encrypting the location data, the user can successfully disable the LS from inferring the original user location and thereby using it for any malicious prospect. For sharing location data with clients, the user can distribute access rights to the users along with the appropriate keys to decrypt the data. In this way, the client can download the desired user's accurate location from the server without the server inferring anything about the downloaded location. The drawback of using encryption based techniques is that the LS cannot perform any operations on the encrypted location data [1]. Therefore, queries such as range queries etc. cannot be performed by the LS.

2.3.5 Position Sharing (PS) [3]

As already discussed, PS is motivated by presence of non-trustable parties in the system. The original user location is broken down into a set of shares each of

which carries equal but limited information about user's original position. These shares can be fused back to recreate the original location with an accuracy depending upon the number of fused shares. The mobile device generates these shares for each position fix and uploads them to a set of LSs; one share per LS. This distribution of shares eliminates single points of failure from the system. From the point of view of any single LS, it holds a degraded user location which makes PS similar to *Location obfuscation*.

When a client LBS wants to query a particular user's location, they can request shares from the LSs for which they have access credentials. These individual credentials are provided to the LBSs by the user. A user can therefore control the level of privacy that they want to maintain with any particular LBS. On fusing these shares, the client can improve the location accuracy to more than that of any one LS. In this respect, PS is similar to *Coordinate Transformation* as the downloaded location can be improved by fusion of further shares.

If a particular LS is compromised as a result of an attack, then the information gain of the attacker is limited to the share of user location hosted by the particular LS. Therefore, this approach provides the graceful degradation of user privacy. With the presence of several LSs in the system, the load of location queries per LS is also decreased. However, communication cost per location update is increased because each LS in the system needs an updated share which corresponds to the current user location.

As optimization of communication in PS is the main focus of this thesis, we will discuss this approach in detail in the next two chapters.

2.4 Attacks on Location Privacy [1]

To better understand the considerations in the design of privacy protection approaches, it is of utmost importance to understand the perspective of an attacker. Many proposed realistic forms of attack on user privacy based on location data have been improvised during the research on privacy protection. These attacks help in the evaluation of a privacy approach by quantifying its robustness against them.

Looking from the bird's eye view, these attacks have been categorized into some major categories in [1]. In this work, the knowledge of attacker has been assumed to vary in two dimensions, i.e. *Contextual* and *Temporal* as shown in Figure 2.4. In the *temporal* dimension, the attacker can either perform a Single Position based attack (Snapshot) or a Multiple Position based attack (History) depending upon the number of user positions they can access. In the *contextual* dimension, the attacker can attack either with no context knowledge by analyzing the available user position(s) or with contextual information, about the user or privacy protection algorithm etc., to reduce or undo the protection implemented by privacy protection scheme.

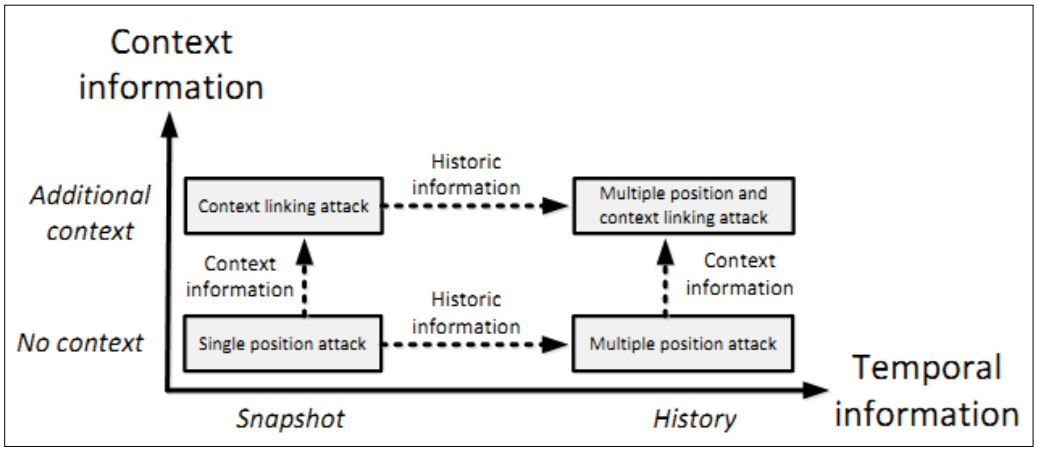


Figure 2.4. Dimensions of attacker's knowledge [1]

Considering this categorization of attacker's knowledge, we will briefly discuss attacks with and without contextual knowledge.

2.4.1 Attacks without contextual knowledge

2.4.1.1 Single Position Attacks

It is possible that the attacker can infer more knowledge, then intended, about exact user location from a single location query. This is possible on the basis of their knowledge of the algorithm. In the case of *k*-anonymity, two such attacks are mentioned in [1]. First is the *Location Homogeneity attack* where lack of diversity in positions of members of a *k*-cluster results in reducing the size of cloaking region for the attacker. The size of cloaking region is also reduced by apply map matching and excluding that area which cannot possibly be occupied by the members of *k*-cluster. The second kind of attack is the *Location Distribution attack*. This attack is effective when one of the *k*-cluster members is located significantly away from the other members and near the boundaries of the cloaking region. The attacker can assume that the isolated member was the one which initiated the formation of the cloak. If one of the other members had initiated cloaking, they would not have gone out of the way to include the isolated member in the cloak as they are already in a densely populated area. The extension of *l*-diversity, if appropriately applied to ensure homogeneous coverage of cloaking region by *k*-cluster members, can be effective against these attacks.

As another example, consider the case of an obfuscation technique that generates obfuscated position with a non-uniform distribution [3]. The attacker can, therefore, reduce the size of obfuscation region by removing areas where probability of finding the user is very low. Here again, obfuscation with homogeneous distribution for degraded position could help to avoid this attack.

2.4.1.2 Multiple Position Attacks

In this case, an attacker uses their knowledge from previous location queries to converge to actual user location. These attacks also use the knowledge of the privacy protection algorithm.

For *k*-anonymity, an intersection between cloaking regions of several queries can help the attacker to associate all requests to a particular user [19]. In the case of location obfuscation, multiple queries for same user location, e.g. user is not moving, will each generate a different obfuscation region [19]. The attacker can use the intersection these regions to narrow down to original user location.

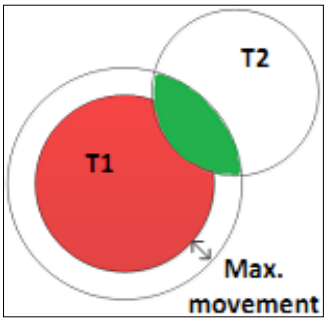


Figure 2.5. Maximum movement boundary attack [1]

Another important means for attacker's help can be the *Maximum Movement Boundary attack* [20]. In this case, the attacker uses the current obfuscation region defining user position, an estimate of user's maximum velocity and the time of the next position update to estimate maximum distance travelled outside the current obfuscation region. The attacker then overlaps this region with the obfuscation region of the next user position. The obfuscation region apart from the intersection can be safely ignored as the user could not have travelled to that region in the given time between the two user positions. This has been illustrated in Figure 2.5.

It has been shown in [5] that location tracks of a person can reveal a lot of personal information about them. In this work, the chances of inferring a person's home location from location tracks have been studied. The coordinates of home location can then be input to look up a name and address using a free Web Service.

2.5 Attacks with contextual knowledge

If the attacker has additional knowledge apart from the location data, they can use this information against the protection provided by the privacy algorithm. Consider for an example that the attacker knows the home address of the user. Now whenever there is the case that on querying the LS, the obfuscation area returned is overlapping the house of the user, the attacker can safely assume that the person is at their home.

Map knowledge is another advanced form of attack for privacy approaches that do not take map into consideration [1]. Suppose here that the attacker has inferred from previous user movements that their victim is driving a car. The natural assumption that the attacker can now take is that the victim is travelling on road networks. Therefore, if *k*-anonymity was being used, the cloaking region can easily be narrowed down to the people travelling on road networks. Same goes for position obfuscation. The obfuscation area returned by the LS can be cut down to leave only the part overlapping with road networks [1]. This is illustrated in Figure 2.6.

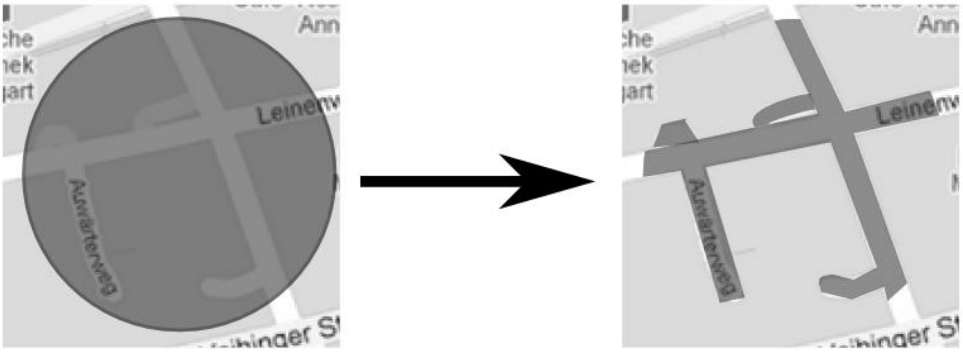


Figure 2.6. Map Knowledge attack: Reduction of obfuscation area to road network [1]

2.6 Other attacks

The attacker can also target the LS directly [1]. If the security measures taken on the LS for protecting location data are not enough, such an attack may be successful. It is also possible that the operator of the LS is corroborating with the attacker e.g. the operator is interested in selling the user data to a third party. All these arguments support the notion of a TTP independent system. On the other hand, a trusted client can also be a single point of failure in the system. If a trusted client gives up exact user location to the attacker or the attacker access the improperly secured user location, the privacy of the user can be lost.

3. System Model and Problem Statement

In this chapter, we will first discuss the system model for PS algorithm and its modifications done to accommodate overhead optimization techniques. In the second part of this chapter, we will define the problem statement in the form of the overhead optimization metrics.

3.1 System Model

In PS approach described in [3], it is assumed that the user's mobile device generates shares for each position fix and uploads them to a set of LSs. As seen in Figure 3.1, this is performed by a trusted *share generation* service running on the device which directly accesses the position generated by the GPS receiver. Each LS receives one share of the current user position.

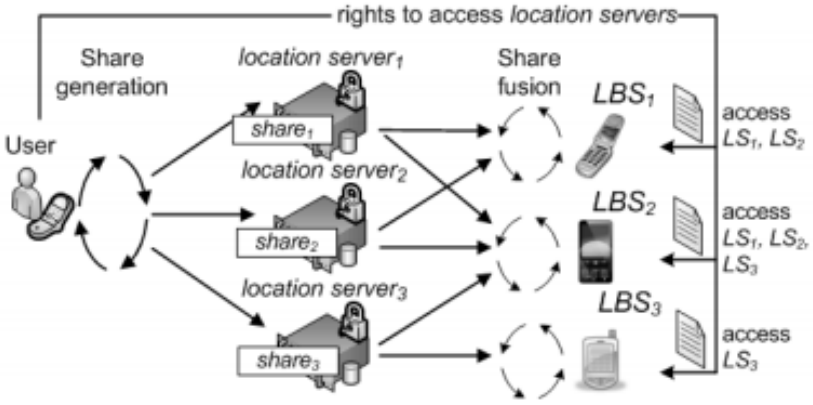


Figure 3.1. Position Sharing: System Model [3]

The querying of shares on a client device is done by a LBS which is a third party software and therefore not trusted. An access control policy is also implemented by every LS whereby a share query is only answered when the client LBS possesses the access rights to the particular share. All LBSs have a *share fusion* component which aggregates the shares to generate a user position of a certain precision.

3.1.1 Formal Definition [3]

As no map knowledge is taken into account, the shape of the obfuscation region in PS is circular. The precision of user location is governed by the radius of the obfuscation region. The larger the radius of the obfuscation region, the lower is the precision of the associated user position. The share generation occurs in such a manner that any share, when fused, is capable of increasing the precision

of calculated position by equal amount. In other words, the fusion of any one share on an imprecise user position will decrease the radius of the obfuscation region of the position by the same amount.

Apart from the original user position, denoted by π, the *share generation* service takes two parameters for its functionality. The first one is the minimum precision, ϕ_{min}, which defines the radius of the obfuscation region for the least precise user position. The second one is the number of shares to be generated, denoted by n. Given ϕ_{min} and n, the *share generation* component's interface is summarized as follows.

$$(s_{master}, \boldsymbol{S}) = generate(\pi, \phi_{min}, n) \quad \text{Eq. 3.1}$$

where $\boldsymbol{S} = \{ \vec{s}_1 , \vec{s}_2 \dots \vec{s}_n \}$

The output returned by the *share generation* component is the requisite shares of limited precision. The part of the output representing the least precise user location is the *master share*, s_{master}. The other output $\boldsymbol{S}$ is the set of, so called, *refinement shares*. All LSs receive s_{master} as well as a single refinement share per location update. For *share fusion*, the availability of s_{master} is a necessary requirement. However, the available number of *refinement shares* may vary. On fusion of a set $\boldsymbol{S}'$of refinement shares, a position π' results whose precision depends upon the number of refinement shares in $\boldsymbol{S}'$.

$$\pi' = fuse(s_{master}, \boldsymbol{S}') \quad \text{Eq. 3.2}$$

If $\boldsymbol{S}'$= $\boldsymbol{S}$, then the precise user location is recreated i.e. , $\pi' = \pi$ with precision, $prec(\pi') = 0$. If a subset of $\boldsymbol{S}$ is fused, then $prec(\pi') > prec(\pi)$.

The refinement shares in $\boldsymbol{S}$ are geometric vectors which are added on top of s_{master} to yield the precise user location. Depending upon the method employed for generation of these vectors, it is possible that the intermediate imprecise user positions generated by fusion of some of the share vectors have a predictable distribution in the associated obfuscation areas. To quantify the effect of this predictability, a privacy metric in the form of the following probability distribution is defined.

$$P_{attack}(\phi) = \Pr (prec(\pi_{attack}) \leq \phi) \quad \text{Eq. 3.3}$$

In order to explain this probability distribution, consider that the attacker has access to a certain k number of shares and they can fuse them to get the position π_k. By knowledge of the *share generation algorithm*, the attacker may generate the remaining $(n - k)$ shares and fuse them on top of π_k to get π_{attack}. If π_{attack} has precision smaller that a user defined distance ϕ, then $P_{attack}(\phi)$ defines the probability for happening of such a case. It is important to note here that the precision of π_{attack} is measured as its distance from the precise user location. The generation of $(n - k)$ shares may be done several times by the attacker to get a probability distribution of precise user location. The privacy metric can therefore quantify for a user that given k shares, what is the probability that an attacker can locate them within a specified distance.

3.1.2 Modifications of System Model for Overhead optimization

We saw in Figure 3.1 that the *share generation* service was responsible for generating and distributing the shares among LSs for each position fix. To

implement a communication overhead optimization policy, we replace this service with *optimized share generation* service as illustrated in Figure 3.2. This new service has an *overhead optimization policy* component and a *share generation* component. The *overhead optimization policy* component has direct access to the position fixes coming from the GPS receiver and controls whether or not to generate new shares using *share generation* component. The *overhead optimization policy* component also handles the distribution of shares among the LSs. The role of this service in terms of user privacy is critical therefore it is assumed to be trusted part of the system.

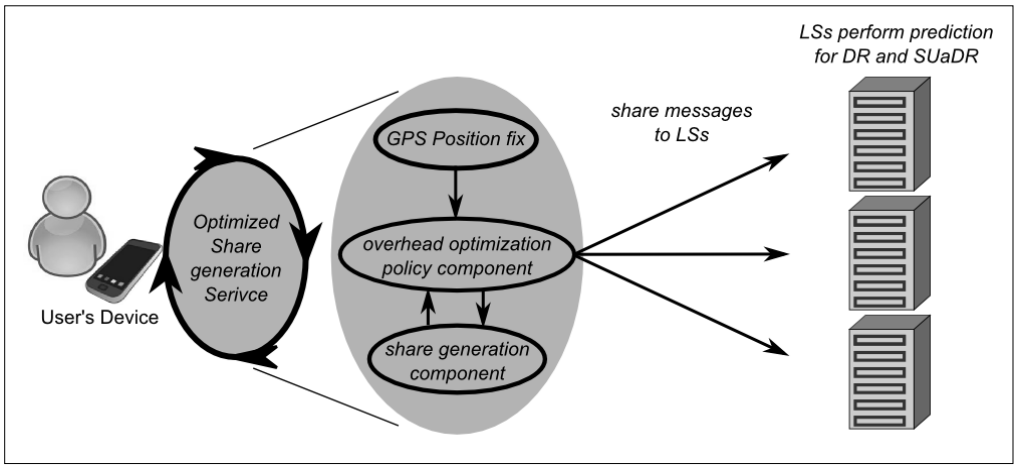


Figure 3.2 Modification in PS's system model for communication overhead optimization.

On the other hand, the LSs are now able to perform prediction of user position in case of DR or SUaDR being the employed overhead optimization policies.

3.2 Problem Statement

A basic implementation of PS incurs considerable overhead. However, this results in a quantifiable protection of user's privacy, measured by $P_{attack}(\phi)$, which is desirable. Stated simply, our objective is to reduce the overall communication while maintaining the privacy protection provided by the original PS algorithm. Taking into account the primary communication optimization techniques that we are exploring, i.e. DR and SU, we need to consider their characteristic effects on user privacy along with their ability to reduce communication overhead. To this end, metrics of evaluation will now be defined.

3.2.1 Communication Overhead optimiztion metrics

For comparing the optimization of performance for a GPS trace, we have defined the Percentage Message Reduction (PMR) metric as follows.

$$PMR = (1 - \frac{x}{n * M}) * 100 \qquad \text{Eq. 3.4}$$

where x is the total number of messages sent to the LSs during the M GPS fixes of the trace and n defines the number of shares generated per position fix. Here $n * M$ gives the total messages that should be sent to the LSs in non-optimized implementation of PS. We will use PMR values for comparison of the different communication optimization schemes with each other.

To compare the optimization with non-privacy aware systems where there is only one message per position fix, we will also use Message Per Position Fix (MPPF) metric defined as:

$$MPPF = x/M \qquad \text{Eq. 3.5}$$

This metric will also give us a general idea of how the amount of communication optimization is effected by a change in number of shares n.

3.2.2 Privacy Security Metrics

As a straight forward way of measuring the effect on the privacy provided by the original PS algorithm, we will use $P_{attack}(\phi)$ to compare the different optimization techniques. However, because SU and SUaDR involve modification of existing shares, it was considered important that another attack based on the average of the k known shares to the attacker should be defined. This will be referred to as $P_{SA-attack}(\phi)$. Where in $P_{attack}(\phi)$, the attacker predicts a precise user position by generating the unknown $(n-k)$ shares by knowledge of the share generation algorithm, in $P_{SA-attack}(\phi)$ these shares are generated as the average of the k known shares.

As the two probabilities represent independent sources of information regarding precise user position, an attacker can combine both of them to get an even more informative distribution. This will be called $P_{combined\ attack}(\phi)$. The desired objective in terms of privacy security will be to at least maintain or improve the privacy security provided by original PS algorithms under all three attacks.

4. Position Sharing Approach

This chapter will discuss in detail the PS approach as described in [3]. Then a brief overview of its further extensions from [21] will be given in section 4.2.

4.1 PS Algorithm

4.1.1 Share Fusion

Figure 4.1 shows an example scenario where shares are fused to lead from p_0 to precise user location $_4$. Here $c_0 = \{p_0, r_0\}$ is master share which has a center position p_0 having absolute location coordinates. Refinement share vectors are added on p_0 to achieve a position of higher precision.

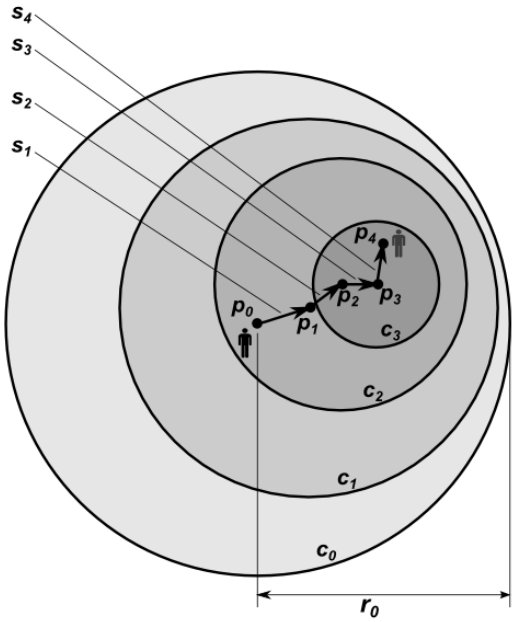


Figure 4.1. Geometric Representation of generated shares for *n* = 4. Original user Position is p_4.

On addition of i share vectors to c_0, its center p_0 will shift to a new more precise position p_i which is the center of obfuscation circle c_i . There is a also a defined decrease in radius of obfuscation circle, denoted by Δ_ϕ, on addition of each share. It is defined as $\Delta_\phi = {}^{r_0}/_{n}$.

When all shares are fused, p_4 is obtained which is the precise user location. Hence no obfuscation circle is associated with p_4. Note here that the order in which the shares vectors are added is not important as addition is a

commutative operation. Therefore, for a given k shares that are fused, there can be $k!$ number of permutations defining the possible orders of share additions. The algorithm of share fusion is summarized in Algorithm 1.

Algorithm 1. Share Fusion Algorithm as given in [3]	
1:	**Function fuse_k_shares**$(\mathbf{n}, \mathbf{c_0}, \vec{s}_1 \ldots \vec{s}_k)$
2:	$\Delta_\phi = r_0/n$
3:	$\vec{p} = p_0$
4:	$r = r_0$
5:	**for** $i = 1$ **to** k
6:	$\vec{p} = \vec{p} + \vec{s}_i$
7:	$r = r - \Delta_\phi$
8:	**end**
9:	**return** $\boldsymbol{c_k} = \{\boldsymbol{p}, \boldsymbol{r}\}$

4.1.2 Share Generation

Given n and r_0 , the problem of share generation involves calculation of a set of vectors $\boldsymbol{S}$ and c_0 for a precise position π such that any permutation of vectors in $\boldsymbol{S}$ can be fused together on top of c_0 to recreate the original position π. For this to be possible, there are certain guarantees that the share generation process should give. These are:

$$p_n \in c_i \quad \text{Eq. 4.1}$$

$$c_i \in c_{i-1} \quad \text{Eq. 4.2}$$

Eq. 4.1 necessitates that each obfuscation circle should contain the precise user location. This means that the probability of finding user location within the obfuscation area must always be nonzero. Eq. 4.2 implies that each obfuscation circle should lie completely with the obfuscation circle of the next imprecise position. From the point of view of share generation, this means that the increase in radius of the obfuscation circle when moving from one position, say p_i, to a more imprecise position, p_{i-1}, should always be greater than the shift vector, s_i, between the two positions. As increase in imprecision between one position to the next imprecise position is desired to be fixed to Δ_ϕ, so we conclude that:

$$|\vec{s}_i| \leq \Delta_\phi \; \forall \; i \in [1, n] \quad \text{Eq. 4.3}$$

Considering these constraints, two share generation approaches have been devised in [3]. We will now discuss these approaches.

4.1.2.1 a-posteriori Share Generation

In this appraoch, the most obfuscated user position $p_0$is generated as a last step of the algorithm. The direction of each share is taken arbitrarily in the 360 degrees. Length of each vector is selected randomly in the range defined by Eq. 4.3. The pseudo code of the algorithm is given in Algorithm 2.

Algorithm 2. a-posteriori Share Generation Algorithm as given in [3]	
1:	**Function gen_n_shares_posteriori**$(\mathbf{r_0}, \mathbf{n}, \boldsymbol{p_n})$
2:	$\Delta_\phi = r_0/n$
3:	**for** $i = 1$ **to** n

4:	select $\vec{s}_i$ randomly where $\lvert\vec{s}_i\rvert \leq \Delta_\phi$
5:	**end**
6:	$\vec{p}_0 = \vec{p}_n - \sum_{i=1}^{n} \vec{s}_i$
7:	**return** $(\vec{p}_0, \vec{s}_1 \dots \vec{s}_n)$

4.1.2.2 a-priori Share Generation

This approach first selects p_0 randomly within c_0 and then tries to compute a set of share vectors which lead from p_0 to the precise position p_n. To increase the chance of finding the right set of shares, $n-1$ shares are chosen randomly as in a-*posteriori* share generation and the last share is calculated by taking the difference between p_n and the concatenated position of the remaining shares on top of p_0. If this last share does not satisfy the condition in Eq. 4.3, then the process is repeated until such a case happens. Otherwise the share generation is complete. This is represented as the pseudo code in Algorithm 3.

Algorithm 3. a-priori Share Generation Algorithm as given in [3]	
1:	**Function gen_n_shares_priori**$(\mathbf{r_0}, \mathbf{n}, \boldsymbol{p_n})$
2:	$\Delta_\phi = r_0/n$
3:	select $p_0$randomly such that $dist(p_0, p_n) \leq r_0$
4:	**while** $dist\left((\vec{p}_0 + \sum_{i=1}^{n-1} \vec{s}_i), p_n\right) > \Delta_\phi)$
5:	**for** $i = 1$ *to* $n-1$
6:	select $\vec{s}_i$ randomly where $\lvert\vec{s}_i\rvert \leq \Delta_\phi$
7:	**end**
8:	**end**
9:	$\vec{s}_n = \vec{p}_n - (\vec{p}_0 + \sum_{i=1}^{n-1} \vec{s}_i)$
10:	**return** $(\vec{p}_0, \vec{s}_1 \dots \vec{s}_n)$

4.2 Extensions of Position Sharing

In another work of the authors of PS [21], they have referred to the above described approach as Open Space Position Sharing with Any Share Order (OSPS-ASO). This is to highlight the fact that the order of share fusion did not matter in this approach. However, in a next two approaches, share order is important. These will be discussed as presented in [21].

4.2.1 Fixed Share Order

This approach is called Open Space Position Sharing with Fixed Share Order (OSPS-FSO). In OSPS-ASO, we saw that the shares length is restricted by Δ_ϕ, therefore the precise user position becomes predictable with the increased number of shares the attacker can access. With this as motivation, OSPS-FSO allows share vector lengths to exceed Δ_ϕ violating the condition defined in Eq. 4.3. The result of this is that obfuscation area of every next precise position may not necessarily be completely contained within obfuscation area of an imprecise position. Therefore, from the two conditions of Eq. 4.1and Eq. 4.2 which were met by OSPS-ASO, only Eq. 4.1 is fulfilled i.e. p_n should lie within all obfuscation regions. This can be seen more clearly in

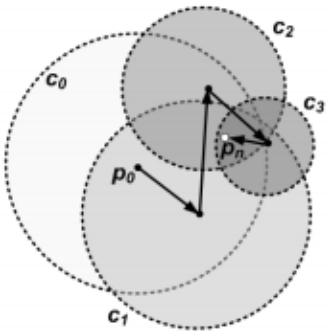


Figure 4.2. OSPS-FSO: Example of shares with unrestricted length [21].

However, the real obfuscation region achieved in this manner is by the overlap of individual obfuscation circles. This might substantially reduce the obfuscation area. The increase in precision of the obtained imprecise position per share is now unequal. On adding a share, the increase in precision of the obtained position will be inversely proportional to overlap of obtained obfuscation area and previous obfuscation region. In order to make the precision increase per share equal, the size of overlapping area has to be increased to become equal to the area of original obfuscation circle, c_i, with radius $(\phi_{min} - i.\Delta_\phi)$. This is achieved by increase in size of the c_i as shown in Figure 4.3.

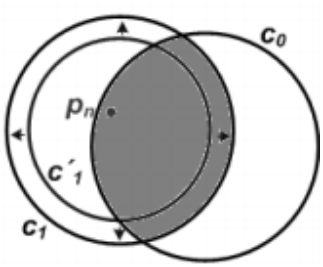


Figure 4.3. $c_1^{'}$ resized to c_1. Overlap area now equals area of $c_1^{'}$ [21].

This increase in radius can be easily undone by the attacker. To make it ineffective, the new obfuscation circle should also be shifted as demonstrated in Figure 4.4. It is important to note here that the shift operation may cause the overlapping area of c_i with c_{i-1} … c_0 to change. Therefore, if the overlapping area is reduced, then an increase in radius is re-performed until the area condition is satisfied.

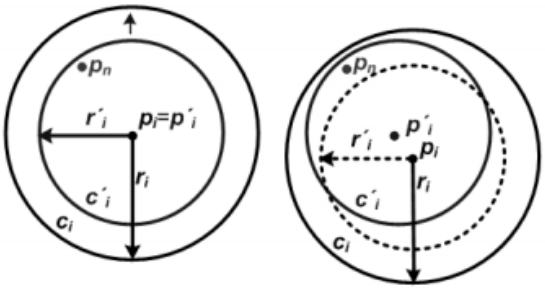


Figure 4.4. Left: Radius of $c_i^{'}$ increased to c_i. Right: p_i shifted randomly while maintaining $c_i^{'}$ within c_i [21].

The results show a significant increase in security, compared to OSPS-ASO, especially when the attacker has access to greater number of shares. A compromise, however, is seen in the decrease in the robustness of the share fusion due to fixed share order. If a LS with one of the intermediate shares is dysfunctional, then positions of higher precision cannot be created.

4.2.2 Map Based CSPS-FSO

If the attacker uses map knowledge, they can significantly reduce the obfuscation area of an imprecise user position as discussed in section 2.5. To this end, the authors in [21] proposed a modified version of OSPS-FSO, called Constrained Space Position Sharing with Fixed Share Order (CSPS-FSO), which takes map knowledge into account. Instead of assuming that the user can move around freely as in OSPS-FSO, CSPS-FSO considers only those areas where the user can be located. For this purpose, the mobile device is assumed to have a binary map which relates whether the user can be present at a particular location or not.

During the process of share generation, the radius increase step from OSPS-FSO as described in Figure 4.3 is modified. The increase of radius of c_i is not stopped when overlapping region's area equals c_i''s area. Instead, it is stopped when the area resulting from the intersection of the overlapped region with the binary map, equals c_i''s area. This is illustrated in Figure 4.5.

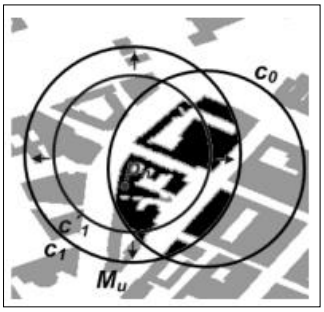


Figure 4.5. Inclusion of Map Knowledge: Black area inside overlapped region equals area of c_1' [21].

To summarize, CSPS-FSO does increase the security of user by avoiding map knowledge based attack. Compared to OSPS-FSO, a price is, however, paid in the form of additional computational overhead for computing intersection area of c_is and the binary map.

In the next chapter, we will discuss the techniques employed for the purpose of reducing communication overhead in the PS approach [3].

5. Optimization of Communication Overhead

As already mentioned, we have taken DR, SU and SUaDR as approaches to optimize location updates from the mobile device to the LSs. In this chapter, we will briefly discuss these techniques and the formulation for their integration with PS.

5.1 Dead Reckoning (DR)

5.1.1 Overview

DR approaches take advantage of predictability of mobile object movement to avoid the need of regular position updates. If the route of a mobile object can be predicted with sufficient accuracy, then the LS can perform the necessary calculations to determine the updated object position for serving any position queries.

In other words, DR is the process of approximation of an object's state based on its last known value and additional information about its rate of change over time. In context of position estimation, the rate of state change can be the velocity of the object or its higher derivatives. To avoid continuous update of object position, an estimate of the object's relative movement can be made by integrating the last known value of its velocity over time and adding it to the last known position to estimate current position.

The detail of application of DR techniques varies with the scenario. For example, consider the case of predicting user position indoors where GPS signals and WiFi positioning are not available [22]. As soon as the user enters an indoor environment, their last known position reported by GPS is kept as an absolute reference. Thereafter, user movements are predicted using mobile device's sensors such as accelerometer, gyroscope and magnetometer etc. These movements are integrated on the principle of DR to estimate the indoor position of the mobile device.

5.1.2 Motivation

DR approaches are famous for minimizing location updates in mobile objects [9]. For the specific scenario of updating a LS (non-privacy aware system) from the mobile device, the LS is the less informed party and the mobile device is assumed to receive a new position fix every few seconds. The mobile device sends updates to the LS so that it can be kept in sync with respect to user

position. Consider for example that the last update packet sent to the LS at time t_{last} was $update_{t_{last}}$ as given in Eq. 5.1.

$$update_{t_{last}} = (\vec{p}_n^{\,t_{last}}, v^{t_{last}}, \vec{u}^{t_{last}}) \quad \text{Eq. 5.1}$$

$$\vec{p'}_n^{\,t} = \boldsymbol{CalcPosition}(update_{last}, t) \quad \text{Eq. 5.2}$$

$$D = |\vec{p}_n^{\,t} - \vec{p'}_n^{\,t}| \quad \text{Eq. 5.3}$$

The update packet included the user position $\vec{p}_n^{\,t_{last}}$, their speed $v^{t_{last}}$ and their direction of motion $\vec{u}^{t_{last}}$. The LS can now predict the user movement by performing some calculation ***CalcPosition*** at a certain time $t \geq t_{last}$ to get estimated user position, as shown in Eq. 5.2. If the user's device also performs ***CalcPosition*** after receiving a new position fix, say p_n^2 at time 2, they can determine the discrepancy D, between the actual reported position and the position being estimated by the LS from Eq. 5.3.

Now the user can set an upper bound on the error in user position reported by the LS by sending a new update to the LS whenever D exceeds a certain desired threshold DR_Th. In this way, a reduction in the number of position updates is be achieved by compromising accuracy of user location returned by the LS to clients. The overall effect of the algorithm is depicted in

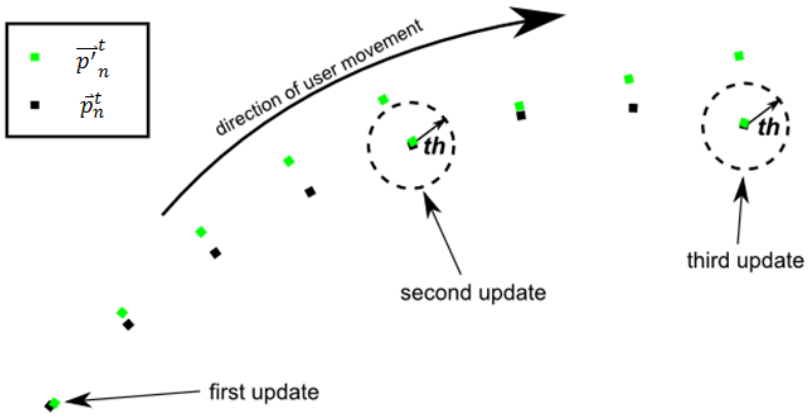


Figure 5.1. Dead Reckoning: Mobile device sends location update whenever predicted position (Green dots) differs from Real Trajectory (Black dots) by more than th.

5.1.3 Background

During the course of research, various implementations of DR have been developed [8, 9]. Here is an overview of some common difference in these approaches.

5.1.3.1 Prediction functions

The prediction function ***CalcPosition*** can vary depending upon the user movement being modeled. For a route that has smooth turns, modeling angular movement along with linear motion can help reduce more location updates rather than considering only the direction of motion during the last position update. As mentioned in [8], higher order prediction functions rather than only linear and angular speeds can result in better modeling of user motion.

Simpler methods of updating the LS also include distance-based or time-based DR whereby the mobile device resends an update to the LS after a certain fixed distance has passed or a certain time has elapsed [8].

5.1.3.2 Threshold variants

Two policies of implementation stem by fixing a constant threshold or adaptively varying it during the course of the trip. In policies with constant threshold, the value of threshold is decided on the basis of available information about the trip such as knowledge of route, average trip speed, affordable inaccuracy in position and desired amount of reduction in communication cost. For example on a jerky trajectory, setting a small value of DR_Th may result in too many location updates. On the other hand, an advantage gained is to have a more accurate reported trajectory as small value of DR_Th is translated to less inaccuracy in user position.

Approaches have been proposed which keep changing DR_Th during the course of a trip. In [9] for example, the value of DR_Th is decided to be a value which minimizes overall information cost predicted to incurred till the next update. The authors also extend the variable threshold approach to address the advanced problem of disconnection detection by the LS.

5.1.3.3 Variants with Map knowledge

By have knowledge of the route followed by the user, the need of predicting movement direction is considerably diminished as implemented in [8]. Therefore, DR is reduced to a problem of predicting user position along a straight line. The variation in linear speed and acceleration etc then determine the number of updates sent by the mobile device. However, additional computational cost is incurred during map matching to locate which route is being taken by the user.

As further explained in [8], an advanced problem in map based protocols is the correct prediction of turns taken by the user at road junctions. It is solved by building a user-specific or general probability distribution over the turns taken at each road junction based on the history of traffic.

5.1.4 Formal Definition

To consider integrating DR for PS, we need to first consider the differences that have arisen in the non-privacy aware system due to PS. Where in a non-privacy aware system, there is a single LS handling user location data, there are n LSs holding position information in PS based systems. Therefore, if dead-reckoning is to be applied, each LS has to predict user position. As already explained, each LS only has s_{master} and one refinement share instead of having an accurate user position. Due to the reason that refinement shares are relative vectors, we have not considered them as part of the prediction function. However, s_{master} defines an absolute, though obfuscated, user position in the form of p_0 on which prediction can be applied.

To optimize communication overhead, we want to reuse a set of refinement shares for as many position fixes as possible. For a single share generation, the resultant vector formed by summation of refinement shares connects p_0^1 and p_n^1

in a rigid fashion. While the refinement shares remain unchanged, a new position fix p_n^2 is fully represented by p_0^2 if p_0^2 is the result of the same shift as was present between p_n^1 and p_n^2. This is illustrated in Figure 5.2.

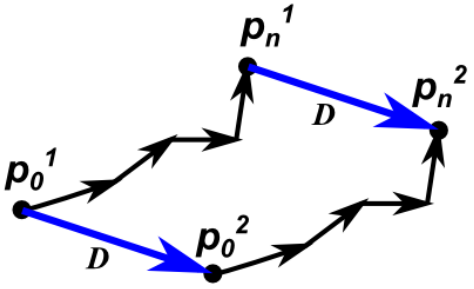


Figure 5.2. Unchanged refinement shares (black arrows): Shift D in p_ns is equal to shift in p_0s.

This implies that a new position for p_0 should be predicted by the LS, fusing shares on top of which should lead to the new p_n. To make the prediction possible using dead-reckoning, update messages sent to each LS are defined as given in Eq. 5.4.

$$update_{t_{last}} = (s_{master}^{t_{last}}, s_r^{t_{last}}, v^{t_{last}}, \omega^{t_{last}}, \vec{u}^{t_{last}}) \qquad \text{Eq. 5.4}$$

Apart from the s_{master} and s_r which are part of the update message as per PS protocol, we also send linear and angular velocities, $v^{t_{last}}$ and $\omega^{t_{last}}$ respectively, and the direction of motion $\vec{u}^{t_{last}}$. The server then uses this additional information to calculate new position of p_0 when a query is made by a client. Any error that appears in the calculation of p_0 is directly reflected in all positions of greater precision calculated as a result of share fusion algorithm. This is because the error in p_0 represents a shift of the rigid connection formed by refinement shares between p_0 and p_k when k shares are fused. In Figure 5.3, p'^2_0 is the predicted position of p_0^2 calculated by the LS using Eq. 5.5 with $t_{now} = 2$ and $t_{last} = 1$. The error e in predication of p'^2_0 has resulted in same error in the fused user position p'^2_n.

$$p'^{t_{now}}_0 = \boldsymbol{CalcPosition}(p_0^{t_{last}}, v^{t_{last}}, \omega^{t_{last}}, \vec{u}^{t_{last}}, t_{now}) \qquad \text{Eq. 5.5}$$

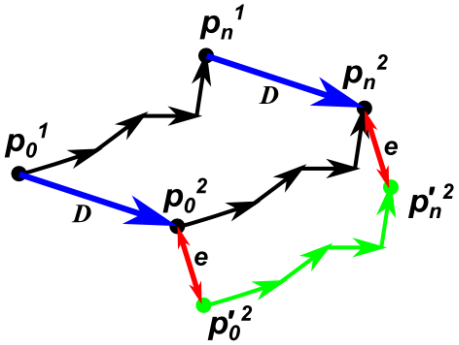


Figure 5.3. Error e in fused position p'^2_n is same as error in p'^2_0 predicted by the LS.

5.1.5 Pseudo code

Algorithm 4 gives the pseudo code of the *optimized share generation* service. This algorithm runs on the device of the user and is considered to be trusted.

Algorithm 4. DR based Optimized Share Generation service	
1:	**while 1**
2:	$p_n^{t_{now}} = wait\_for\_next\_position\_fix$
3:	**If** $exists(update_{t_{last}})$
4:	$p'^{t_{now}}_0 = \boldsymbol{CalcPosition}(p_0^{t_{last}}, v^{t_{last}}, \omega^{t_{last}}, \vec{u}^{t_{last}}, t_{now})$
5:	$s'^{t_{last}}_{master} = (p'^{t_{now}}_0, r_0)$
6:	$p'^{t_{now}}_n = fuse(s'^{t_{last}}_{master}, \boldsymbol{S^{t_{last}}})$
7:	**If** $dist(p_n^{t_{now}}, p'^{t_{now}}_n) > DR\_Th$
8:	$(s_{master}^{t_{now}}, \boldsymbol{S^{t_{now}}}) = generate(p_n^{t_{now}}, \phi_{min}, n)$
9:	**for** $i = 1{:}n$
10:	$update_{t_{last}} = (s_{master}^{t_{now}}, s_i^{t_{now}}, v^{t_{now}}, \omega^{t_{now}}, \vec{u}^{t_{now}})$
11:	**end**
12:	**end**
13:	**else**
14:	$(s_{master}^{t_{now}}, \boldsymbol{S^{t_{now}}}) = generate(p_n^{t_{now}}, \phi_{min}, n)$
15:	**for** $i = 1{:}n$
16:	$update_{t_{last}} = (s_{master}^{t_{now}}, s_i^{t_{now}}, v^{t_{now}}, \omega^{t_{now}}, \vec{u}^{t_{now}})$
17:	**end**
18:	**end**
19:	**end**

The algorithm starts by waiting for a fresh position fix from the GPS sensor. New shares are generated only for the first position fix (line 13-18) or when predicted user position $p'^{t_{now}}_n$ differs from actual user position $p_n^{t_{now}}$ by more than DR_Th units of distance (line 7-12). For each new position fix at time t_{now}, the user's device performs ***CalcPosition*** function to determine the current position of master share $p'^{t_{now}}_0$ as predicted by the LSs (line 4). This predicted master share $p'^{t_{now}}_0$ is then used to determine the predicted value of precise user position $p'^{t_{now}}_n$ (line 6). In line 7, it is checked whether the error in the server's estimate is big enough to exceed DR_Th. If such a case happens, the LSs are re-updated with a set of new shares (line 8 - 11).

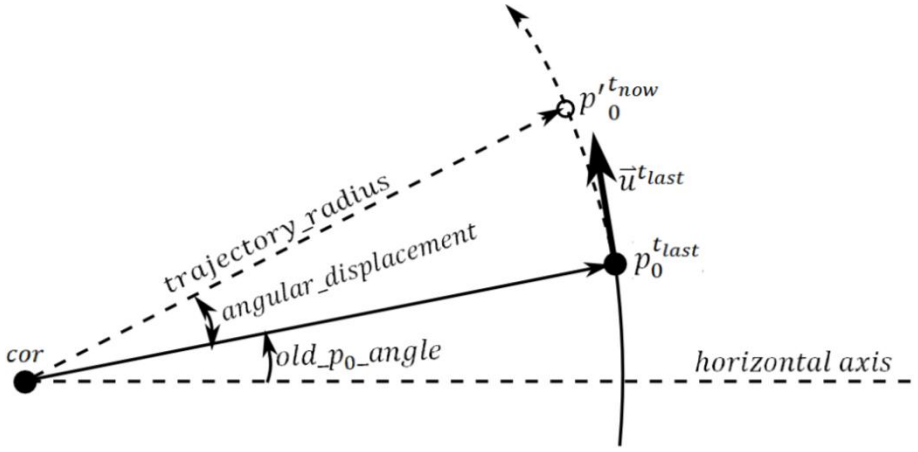


Figure 5.4. Trajectory Prediction considering angular and linear velocities

The LSs answer location queries by estimating a fresh value of p_0 using the ***CalcPostion*** function as done by user's mobile device. As mentioned earlier, our implementation of the prediction algorithm uses angular and linear speeds of the user to predict the future trajectory (see Figure 5.4 and Algorithm 5). If the angular velocity is significant e.g. greater than 1 deg/sec (line2), only then a trajectory with certain radius of curvature is predicted. Otherwise, a straight line trajectory is predicted using only linear speed of the mobile user (line 9-12). When considering angular velocity, radius of curvature of the trajectory is calculated as a first step (line 3). Next, the center of rotation of the mobile user's path is calculated according to radius of curvature, the last known master share position and the direction of motion (line 4). According to the elapsed time between query time and last update's time, angular displacement along the curvature of trajectory is computed and used to estimate $p'^{t_{now}}_0$ in lines 5-8.

	Algorithm 5. Prediction function
1:	**Function** ***CalcPosition***$\left(p_0^{t_{last}}, v^{t_{last}}, \omega^{t_{last}}, \vec{u}^{t_{last}}, t_{now}\right)$
2:	**If** $\omega > 1\ deg/sec$
3:	$trajectory\_radius = v^{t_{last}}/\omega^{t_{last}}$
4:	$cor = calculate\_cor\left(trajectory\_radius, p_0^{t_{last}}, \vec{u}^{t_{last}}\right)$
5:	$angular\_displacement = \omega^{t_{last}} * (t_{now} - t_{last})$
6:	$old\_p_0\_angle = angle(\overrightarrow{p_0^{t_{last}} - cor})$
7:	$p'^{t_{now}}_0 = cor + trajectory\_radius * (angular\_displacement +$
8:	$old\_p_0\_angle\ )$
9:	**else**
10:	$linear\_displacement = v^{t_{last}} * (t_{now} - t_{last})$
11:	$p'^{t_{now}}_0 = p_0^{t_{last}} + linear\_displacement.\vec{u}^{t_{last}}$
12:	**end**
13:	**return** $p'^{t_{now}}_0$
14:	**end**

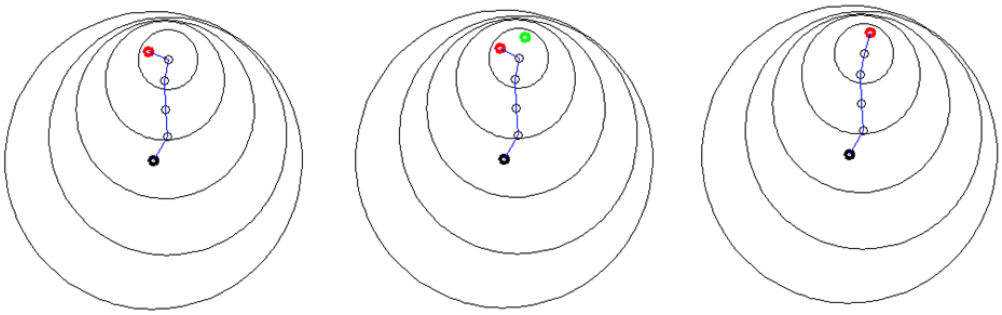
5.2 Selective Update (SU)

5.2.1 Overview

This approach also tries to reuse refinement shares in successive position updates. However, in contrast to DR, SU does not generate a complete set of new shares every time it needs to update the LSs. Instead, on arrival of a new position fix, only minimum number of shares are re-generated and their corresponding LSs are updated so that the user position hosted by the LSs remains consistent with the actual user position. Hence, the last p_0 sent to the LSs is used as long as the SU of some shares keeps successfully representing the new user position. Reduction is communication overhead is helped in two ways. Firstly, s_{master} is reused and therefore not sent to the LSs thus reducing message size. Secondly, the number of shares that are sent to the server vary from *1* to *n* depending upon the course of trajectory resulting from user movements. The details of the algorithm will now be elaborated. The geometric representations of shares used in the following illustrations are on the lines of Figure 4.1 but with $n = 5$.

5.2.2 Motivation

Consider as an example scenario shown in Figure 5.5. Part (a) of the figure shows $p_n^{t_1}$ in red and one possible fusion of its shares along with the obfuscation circles. In part (b), the next position fix, $p_n^{t_2}$ shown in green, arrives close to $p_n^{t_2}$. Using the principle of SU, it can be seen that this small movement from $p_n^{t_1}$ to $p_n^{t_2}$ can be accommodated by a single update of share s_5 as shown in part (c) of the figure. This is because the new position $p_n^{t_2}$ lies within the obfuscation circle c_5 inside which s_5 can be defined freely. Therefore, only one message to the LS which is storing the older version of s_5 will suffice in order to keep the user position with LSs consistent with the new position fix $p_n^{t_2}$.



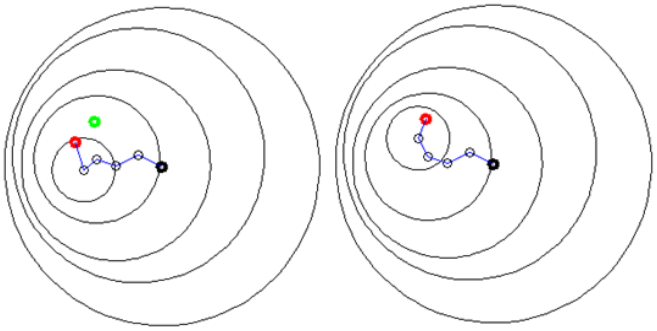
(a) (b) (c)

Figure 5.5. Selective Update Example 1: a) Position fix $p_n^{t_1}$ and its shares, $n = 5$. b) Second Position fix, $p_n^{t_2}$ (Green), appears close to $p_n^{t_1}$. c) s_4 updated to represent $p_n^{t_2}$.

Next we consider another example, Figure 5.6(a), where $p_n^{t_2}$ is a bit further away as compared to the last example. Here, $p_n^{t_2}$ is not reachable even if we re-define s_5 because it lies inside c_3 instead of c_4. A modification and update of two shares vectors s_4 and s_5 make the rest of the shares consistant. A possible redefinition of these shares is shown in Figure 5.6(b).

By these examples, we note for n shares, if the new position fix $p_n^{t_2}$ lies in c_k for $k \leq n$, then atmost $(n - k + 1)$ shares are required to be redefined and updated

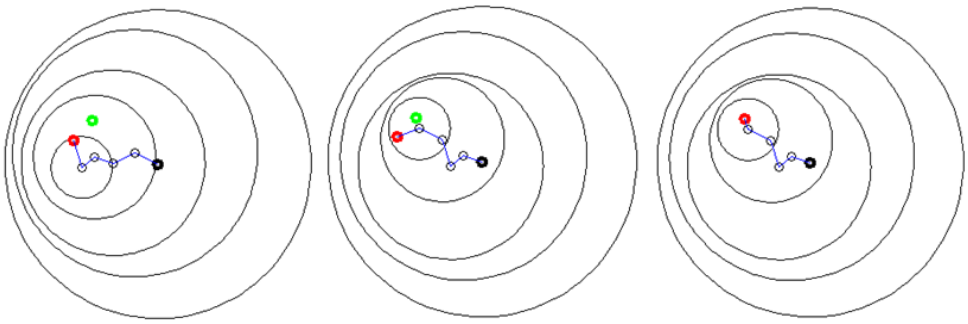
to make $p_n^{t_2}$ consistent. It is also noticeable that at least one share update is required per each new position fix that is different from the previous one.



(a) (b)

Figure 5.6. Selective Update Example 2. a) $p_n^{t_1}$ (Red) and its shares, $p_n^{t_2}$ shown in green. b) s_4 and s_5 redefined to represent $p_n^{t_2}$

In Figure 5.7, the same example is reconsidered. This time however, we change the order of share combination before performing the update of shares. By doing so, it can be seen in Figure 5.7(b) and (c) that it is now possible to reach $p_n^{t_2}$ by changing s_5 only. In comparison to the example of Figure 5.6, a change in share order has allowed us to avoid update of an additional share for the same scenario.



(a) (b) (c)

Figure 5.7. Selective Update with re-ordered shares. a) $p_n^{t_1}$ (Red) and its shares, $p_n^{t_2}$ shown in green. b) Re-ordering of shares of $p_n^{t_1}$. c) s_5 re-defined to represent $p_n^{t_2}$

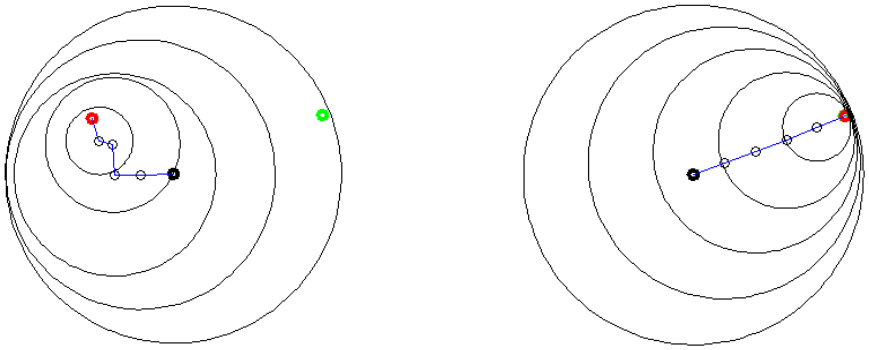
5.2.3 Formal definition

5.2.3.1 Condition for Applicability

Now we will consider the details involved in the application of SU. First of all, the limits of application of this approach are to be defined between consecutive position fixes. This is primarily determined by the fact that whether the new position fix, $p_n^{t_2}$, is reachable from the $p_0^{t_1}$ using n share vectors. It should be noted here that the share vectors $s_1 \ldots s_n$ are modifiable with length of at most $\Delta\phi$ and an arbitrary direction. Therefore, we can summarize this condition for determining the possibility of application of SU as:

$$distance(p_0^{t_1}, p_n^{t_2}) < r_0 \qquad \text{Eq. 5.6}$$

The above condition covers even the case where all n shares need to be updated because the $distance(p_0^{t_1}, p_n^{t_2})$ is almost equal to r_0. An example of such a case is presented in Figure 5.8.

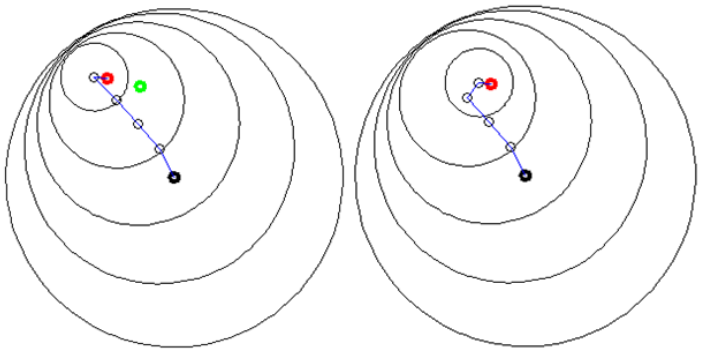


(a) (b)

Figure 5.8. Illustration of n share update. a) First position fix $p_n^{t_1}$ shown in red and its shares starting from $p_0^{t_1}$ (black). Second Position fix $p_n^{t_2}$ shown in green. b) All five shares have been changed in order to represent the new position fix while keeping $p_0^{t_1}$ same.

5.2.3.2 Share Update

Having decided when to apply SU, we need to decide which refinement shares should be updated so that the overall share updates are kept to a minimum. For this purpose, the shares are sorted in descending order of their potential of changeability. As an example, we could consider the length of the shares as a way of determining potential. The share with the least length would have the highest potential because it had the least contribution in leading from p_0 to p_n in terms of distance.



(a) (b)

Figure 5.9. Share Update Example: a) Smallest share s_5 cannot be changed to reach the new position fix (Green). b) Instead, s_4 is changed.

However, this might not be the optimal way of minimizing shares updates as illustrated in Figure 5.9. In part (a) of the figure, we notice that s_5 is the smallest share but it cannot be extended to the new position fix as the new position fix lies outside c_5. However, by modifying s_4 not only in length but also in its direction, consistency for new position fix is achieved as shown in part (b) of the figure. Therefore, it is important that both the length and direction of the shares be taken into account for determining the shares whose re-definition will result in optimally reduced update messages to the LSs.

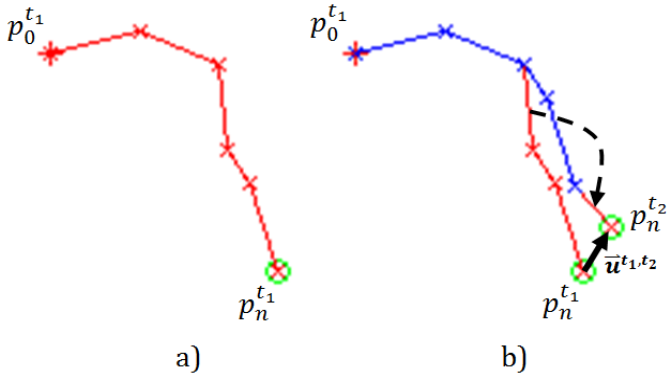


Figure 5.10. Share Update: detailed Illustration: New shares are shown in red, reused are blue. a) Shares generated for $p_n^{t_1}$. b) One share updated because of its least contribution along $\vec{u}^{t_1,t_2}$.

To formulate a systematic way of finding a share's potential, the unit vector in the direction of movement trajectory is considered. As shown in Figure 5.10 b), $\vec{u}^{t_1,t_2}$ denotes the unit vector. Considering that the shares are pointing from $p_0^{t_1}$ towards $p_n^{t_1}$, $\vec{s}_3$ is opposing the direction of $\vec{u}^{t_1,t_2}$ with the largest value of dot product as compared to other shares. Therefore, when the new position fix $p_n^{t_2}$ arrives, $\vec{s}_3$ is the first share to be updated as shown by the dotted arrow in part (b) of the figure.

5.2.4 Pseudo code

The complete pseudo code for the concept is presented in Algorithm 6. After getting a new position fix, complete set of new shares is generated in lines 3-8 if required. Either the shares are created for the first position fix or when the distance between current position fix and last p_0 exceeds a certain threshold SU_Th. The upper limit of SU_Th is defined by Eq. 5.6.

If a complete set of new shares are not required, old shares are sorted in ascending order of their contribution against the direction of user motion (lines 10-11). After this, the number of shares required to be updated, $shares\_to\_update$, in order to reach the new position $p_n^{t_{now}}$ are determined in lines 12-24 starting from the least contributing share. The reasoning behind this is that each share can travel a maximum length of $\Delta\phi$. For example, if two share was being updated for $n = 5$, than distance from $p_3^{t_{last}}$ to $p_5^{t_{now}}$ should be less than 2* $\Delta\phi$. Therefore if distance $dist\left(p_{(n-i)}^{t_{last}}, p_n^{t_{now}}\right)$ is less than $i * \Delta\phi$, then i shares are required to be updated. The shares are then updated to have same length and direction as given in lines 25-29 of the pseudo code.

Algorithm 6. SU based Optimized Share Generation service	
1:	**while 1**
2:	$p_n^{t_{now}} = wait\_for\_next\_position\_fix$
3:	**If** $isempty(last\_shares)$ || $dist\left(p_n^{t_{now}}, last\_shares.s_{master}^{t_{now}}.p_0\right) > SU\_Th$
4:	$\left(s_{master}^{t_{now}}, \boldsymbol{S}^{\boldsymbol{t_{now}}}\right) = generate\left(p_n^{t_{now}}, \phi_{min}, n\right)$
5:	$last\_shares = \left(s_{master}^{t_{now}}, \boldsymbol{S}^{\boldsymbol{t_{now}}}\right)$

```
        for i = 1:n
            update_{t_last} = (s_master^{t_now}, s_i^{t_now})
        end
    else
        u^{t_last,t_now} = unit_vector( p_n^{t_now} − p_n^{t_last} )
        sort_by_contribution(last_shares.S, u^{t_last,t_now} )
        for i = 1:n
            if i == n
                p_{(n−i)}^{t_last} = last_shares.s_master.p_0
            else
                p_{(n−i)}^{t_last} = fuse(last_shares.s_master, last_shares.S((i + 1):n))
            end
            distance_to_be_covered = dist(p_{(n−i)}^{t_last}, p_n^{t_now})
            if distance_to_be_covered < Δφ * i
                start_point = p_{(n−i)}^{t_last}
                shares_to_update = i
                break;
            end
        end
        average_update_vector_length = distance_to_be_covered / shares_to_update
        share_angle = angle(p_n^{t_now} − start_point )
        u_share = unit_vector(share_angle)
        for i = 1: shares_to_update
            last_shares.S(i) = average_update_vector_length.u_share
        end
    end
    p_n^{t_last} = p_n^{t_now}
end
```

5.3 SUaDR

5.3.1 Motivation

The motivation for SUaDR came from observing the behavior of SU and DR algorithms after their implementation. Although the detailed results of these algorithms will be discussed in Chapter 6, it is important to show a few results here to establish the reasoning behind SUaDR.

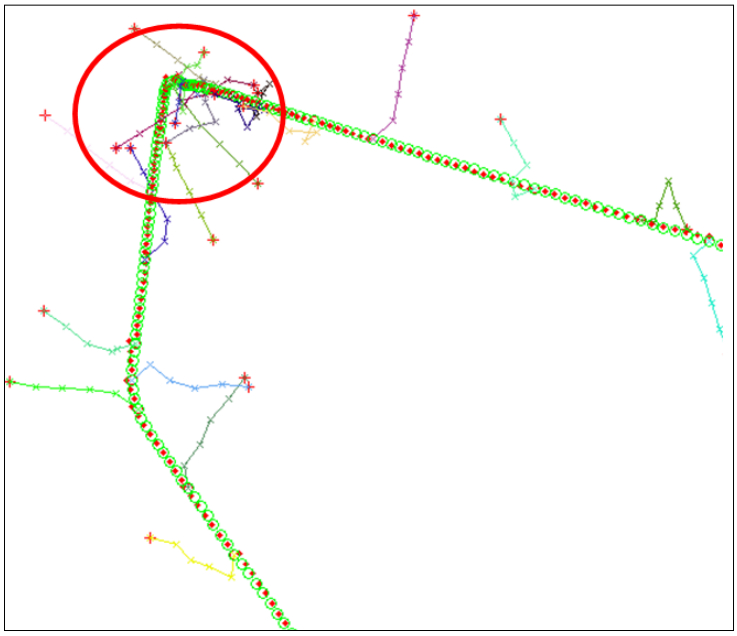


Figure 5.11. Dead Reckoning applied on a test GPS trace (green). The Red circle marks a turn where, clearly, share generations ($n = 5$) become dense compared to straight patches of trajectory.

Figure 5.11 shows that DR is not effective in parts of trajectory where prediction function performs poorly such as turns. The generated shares are shown as connected straight lines joining some of the trajectory points. During sharp turns, DR predicts user position less accurately and thus generates new sets of shares more often resulting in a sharp increase in number of messages results. On the other hand, during straight portions of the trajectory, DR reduces the updates very efficiently. On the other hand, it is also noticeable in Figure 5.12 for example, that SU has the potential of performing consistently during turns as well as straight patches of the trajectory. A plausible idea that can be inferred from this discussion is the merging of the two techniques, i.e. DR and SU, such that strengths of both are reflected in the merger. This merger is called Selective Update and Dead Reckoning (SUaDR).

5.3.2 Formal Definition

We have already seen in section 5.1.4 that DR does not modify the refinement shares for reducing communication overhead. Therefore, it does not significantly affect the privacy security provided by the original share generation algorithm as will be verified later. If optimization of communication overhead achieved by DR can be improved by addition of SU while maintaining the security performance, then it would be an added advantage. To this end, SUaDR tries to achieve this by running DR as the default algorithm and activating SU in parts of trajectory where DR may incur more communication cost.

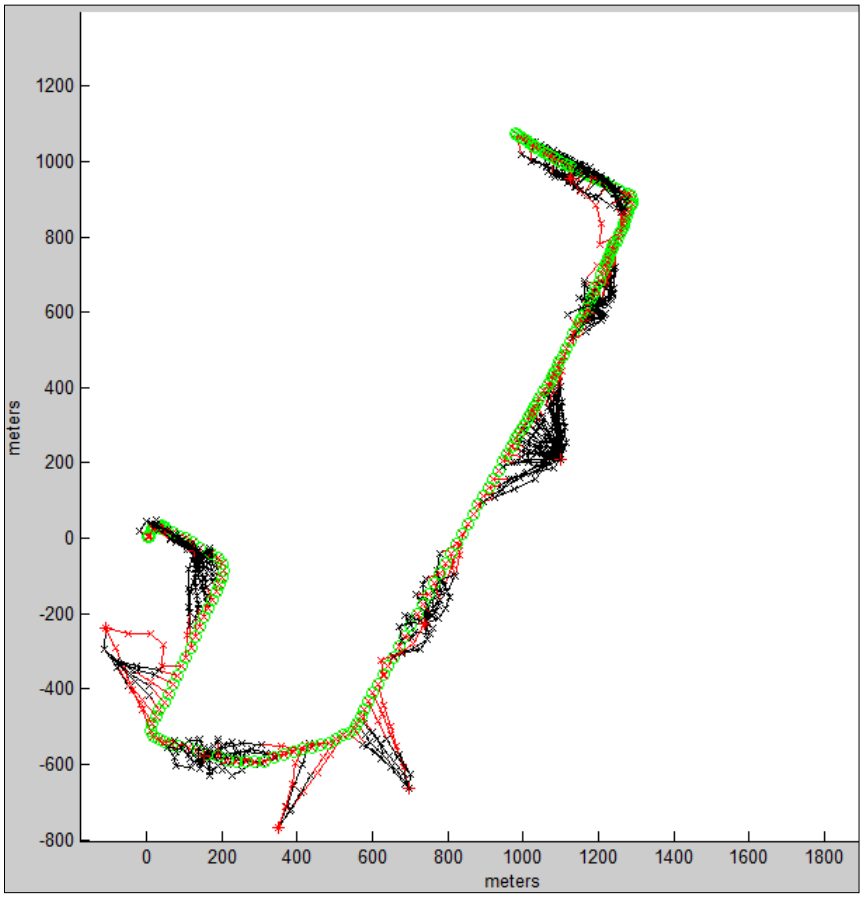


Figure 5.12. SU with $n = 5$, $r_0 = 300$: new shares are red, re-used shares are black. 9 share generations for 400 position fixes (green).

We know that SU sends at least a single update message per position fix. If a new set of shares is sent to the LSs using SU at t_0, then the next n position fixes coming at $t_1 \ldots t_n$ will cost the following number of messages.

$$SU\_messages_{t_1 \ldots t_n} \geq n \qquad \text{Eq. 5.7}$$

Similarly, we also know that DR generates a complete set of shares each time it updates the LSs. If DR updates the LSs k times in n position fixes, then number of total messages sent to the servers will be:

$$DR\_messages_{t_1 \ldots t_n} = k * n \text{ where } k = [0 \ldots n] \qquad \text{Eq. 5.8}$$

The two equations, Eq. 5.7 and Eq. 5.8, show that DR may incur more update messages if multiple updates are performed in n position fixes. This is because the message increase for DR is in multiples of n per update while SU may perform an update with $1 \ldots n$ messages. We will use this observation as a criterion for activating SU in SUaDR.

5.3.3 Pseudo Code

The pseudo code for SUaDR is almost the same as Algorithm 4 except for inclusion of SU as described above. It is given in Algorithm 7 with the added statements highlighted in gray.

Algorithm 7. SUaDR pseudo code

```
1:  while 1
2:      p_n^{t_now} = wait_for_next_position_fix
3:      total_position_fixes = total_position_fixes + 1
4:      If exists(update_{t_last})
5:          p'_0^{t_now} = CalcPosition(p_0^{t_last}, v^{t_last}, ω^{t_last}, u⃗^{t_last}, t_now)
6:          s'^{t_last}_master = (p'_0^{t_now}, r_0)
7:          p'_n^{t_now} = fuse(s'^{t_last}_master, S^{t_last})
8:          If dist(p_n^{t_now}, p'_n^{t_now}) > DR_Th
9:              If (total_position_fixes − last_update_fix < n)
10:                 perform_SU
11:             else
12:                 (s_master^{t_now}, S^{t_now}) = generate(p_n^{t_now}, φ_min, n)
13:                 for i = 1: n
14:                     update_{t_last} = (s_master^{t_now}, s_i^{t_now}, v^{t_now}, ω^{t_now}, u⃗^{t_now})
15:                 end
16:             end
17:         end
18:     else
19:         (s_master^{t_now}, S^{t_now}) = generate(p_n^{t_now}, φ_min, n)
20:         for i = 1: n
21:             update_{t_last} = (s_master^{t_now}, s_i^{t_now}, v^{t_now}, ω^{t_now}, u⃗^{t_now})
22:         end
23:         last_update_fix = total_position_fixes
24:     end
25: end
```

After a brief description of implementation in the next chapter, the detailed evaluation of the SU, DR and SUaDR will be done with respect to the achieved optimization of communication overhead as well as their effects on the security provided by the share generation algorithms.

6. Implementation and Evaluation

In order to evaluate our overhead optimization algorithms on real world GPS traces, we needed an implementation of the original PS algorithm on top of which these techniques could be tested. In the first part of this chapter, I will briefly explain my implementation of the PS, DR, SU and SUaDR. For later comparison, the security analysis of the PS algorithms will be presented as discussed in [3] with addition of our further evaluations with SA attack. Then after a brief description about the GPS traces, we will present the evaluation of each overhead optimization algorithms, first individually and then comparatively.

6.1 Implementation

All algorithms were implemented in MATLAB [23]. Throughout this chapter, the probability distributions shown are a result of Monte Carlo simulations.

6.1.1 PS

The *share generation* and the *share fusion* algorithms from [3], as described in section 4.1 were implemented accordingly. However, for the case of *a-priori* share generation (see Algorithm 3), it was observed that share generation was taking unreasonable amount of time. This was due to the fact that a lot of attempts were required before a set of refinement shares could be found that connected p_0 and p_n (lines 4-8). To overcome this problem, I modified line 6 of the algorithm such that the generation of a refinement share was no more fully random in a circle of radius $\Delta\phi$. Instead, only a small region of this full circle was considered which limited the range and angle for random selection of share.

Starting from p_0, shares were generated to traverse the distance towards p_n. The range of angle and length for generation of a kth share depended upon the distance $Dist_k$. This distance defined a minimum displacement from the p_{k-1} in the direction of p_n that must be travelled by this kth share in order to keep possibility of connecting the preselected p_0 to p_n alive. The computation of $Dist_k$ is defined by the following equations.

$$Dist_{remaining} = distance(p_{k-1}, p_n) \quad \text{Eq. 6.1}$$

$$Dist_{coverable} = (n-k) * \Delta\phi \quad \text{Eq. 6.2}$$

$$Dist_k = Dist_{remaining} - Dist_{coverable} \quad \text{Eq, 6.3}$$

Eq. 6.1 defines the remaining distance that is to be covered collectively by the current share, i.e. kth share, and the rest of $(n-k)$ shares yet to be generated in the direction leading from p_{k-1} to p_n. If all of the $(n-k)$ shares are aligned along this direction with a maximum possible length of $\Delta\phi$, this gave the

maximum distance that was coverable by the rest of the shares as given in Eq. 6.2. Therefore, the kth must at least travel $Dist_k$ (Eq, 6.3) of displacement from p_{k-1} towards p_n. This distance determined the range of angle and length of kth refinement share vector by defining a region of the circle of radius $\Delta\phi$ centered at p_{k-1}. The region is such that a vector originating from p_{k-1} to any point in the region was would have at least a displacement of $Dist_k$ in the direction of p_n. This is illustrated in the cases shown in Figure 6.1.

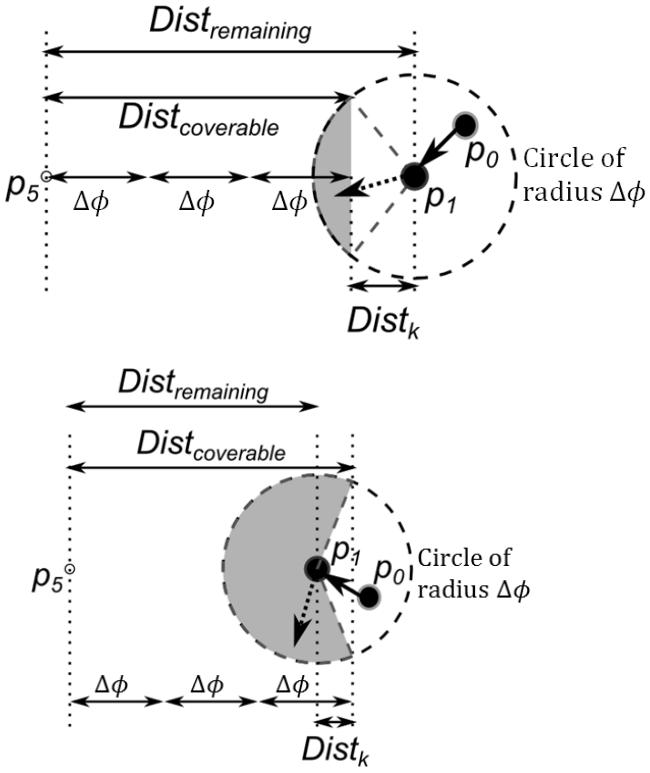


Figure 6.1. *a-priori* share generation with $n = 5$: region for selecting refinement share s_2. Top) $Dist_k$ is positive, small region (grey) left for choice of s_2(dotted arrow). Bottom) $Dist_k$ is negative, more than half of the circle (grey region) available for choosing s_2.

6.1.2 DR

The implementation was based on Algorithm 4. When tested with a randomly chosen GPS trace, the dead-reckoning approach gives promising results. As shown in Figure 6.2, the number of share generations are considerably less as compared to the number of trajectory points p_n^ts. This improvement is despite the fact that the threshold DR_Th has been kept small i.e. almost equal to the distance between two consecutive p_n^ts. A complete new set of shares is generated whenever the predicted value of user position, i.e. $p_n'^t$, differs from corresponding actual user position, p_n^t, by more than DR_Th.

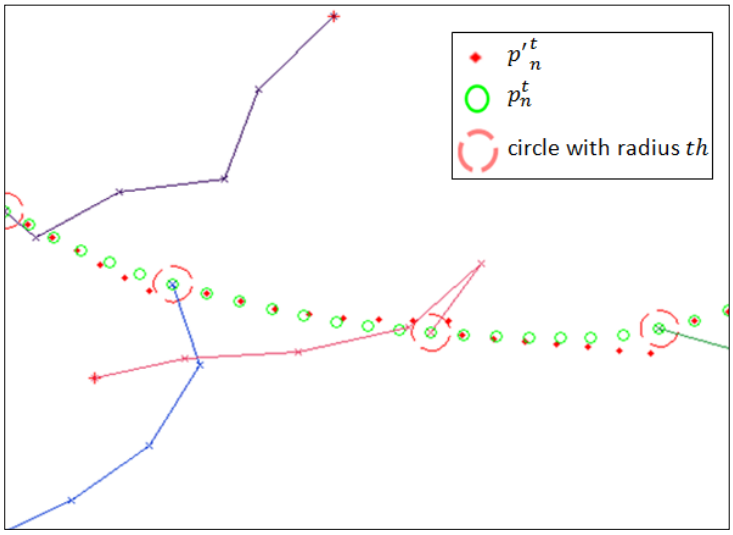


Figure 6.2. Reduced share generations by application of dead-reckoning

6.1.3 SU

Algorithm 6 was implemented as stated in sub-section 5.2.4. We have seen earlier that SU modifies some of the shares to represent the new position fix where possible. Over a set of position fixes, this causes the refinement shares to become stretched in the direction of original user position which motivates the $P_{SA-attack}(\phi)$. This effect is illustrated in Figure 6.3 where a single share generation is being reused over several position fixes. As p_n starts moving away from the p_0, share vectors become more and more aligned in the direction of the vector leading from p_0 to p_n.

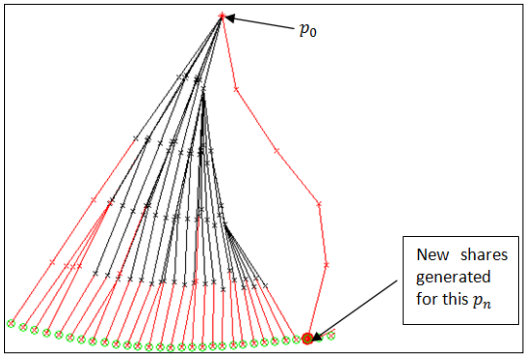


Figure 6.3. Share average attack on Selective Update: reused shares are in black, new shares in Red. Moving from right to left, share vectors are stretched in the direction of the incoming p_ns (green).

6.1.4 SUaDR

The pseudo code of SUaDR as given in Algorithm 7 was implemented. The results of the implementation will be discussed later in this chapter.

6.2 Security Analysis of PS approach

In PS algorithm, the different levels of privacy are provided by the varying size of obfuscation region. If a client has less shares, they should only be able to create a

position with a large obfuscation region compared to another client who has access to more shares. In the obfuscation region defined by the imprecise position p_k, created by fusion of k shares, a client should be unsure about user location with the region c_k i.e. they would assume the precise user position p_n to be uniformly distributed.

However, it was shown in [3] that, an attacker could develop a distribution of user position. This was achieved by generating the unknown shares by knowledge of the algorithm. Using Monte-Carlo simulation, it was verified that such an attack could help the attacker with k shares to find the distribution $P_{attack}(\phi)$ i.e. a non-uniform distribution for p_n in c_k.

6.2.1 Analyzing *a-posteriori* Share Generation

As already seen in *a-posteriori* share generation, p_0 is the resultant position due to summation of pre-generated refinement shares. Therefore the distribution of p_n in c_0 is dependent on the distributions of the summed up refinement share vectors. This distribution was verified by performing a Monte Carlo simulation to be a normal distribution in [3] as shown in Figure 6.4.

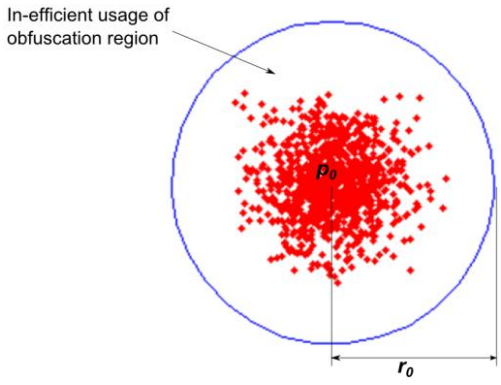


Figure 6.4. User position's distribution in *a-posteriori* share generation [3]: p_n (Red) in r_0 radius around p_0 using Monte Carlo simulation with n = 5 and 1000 runs.

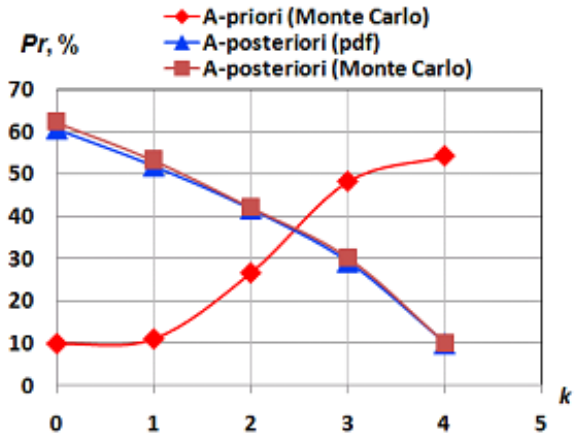
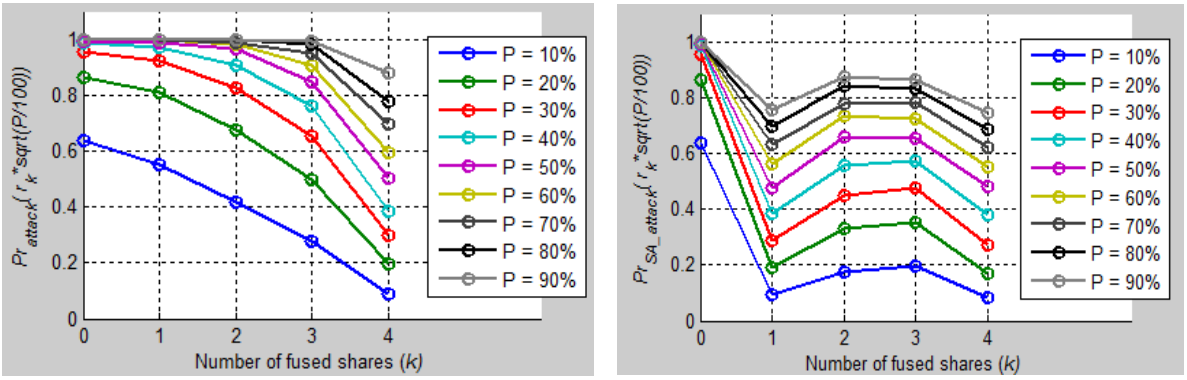


Figure 6.5. Probability of p_n lying with 10% area of c_k around p_k [3].

Similarly, for an attacker having k shares can generate the remaining $(n-k)$ shares randomly and perform this procedure a number of times to obtain a

distribution of p_n around p_k. $P_{attack}(\phi)$ quantifies this threat to user privacy by measuring the probability of an obtained p_n being the actual user position. For ϕ being the radius of 10% of the obfuscation region, the result of Monte-Carlo simulation from [3] are visualized in Figure 6.5 for k fused shares. The graph shows that the efficient usage of obfuscation region increases as the number of fused shares increase, reaching 10% for 4 shares which is equivalent to uniform distribution.

Our implementation also gave the same results (see Figure 6.6(a) for P = 10%). The $P_{SA-attack}(\phi)$ is plotted in part (b) of the figure. The overall effect is seen in Figure 6.7 in the form of $P_{combined\ attack}(\phi)$. As $P_{combined\ attack}(\phi)$ is not very different from $P_{attack}(\phi)$, this demonstrates that he $P_{SA-attack}(\phi)$ was not effective against *a-posteriori* share generation.



(a) (b)

Figure 6.6. *a-posteriori* share generation for n = 5. a) $P_{attack}\ (\phi)$. b) $P_{SA-attack}\ (\phi)$.

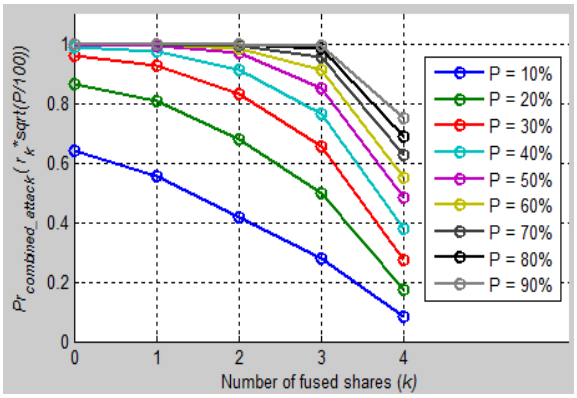


Figure 6.7. $P_{combined\ \ attack}\ (\phi)$ for *a-posteriori* share generation

6.2.2 Analyzing *a-priori* Share Generation

In *a-priori* share generation, selection of p_0 inside c_0 is done before calculation of refinement share vectors. Therefore the distribution of p_n in c_0, or conversely p_0 in c_0, is uniform [3] as shown in Figure 6.8.

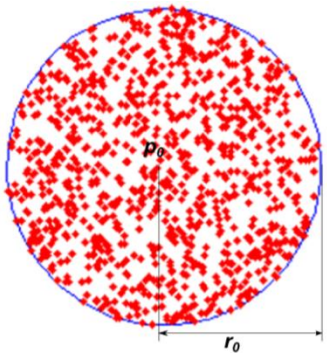
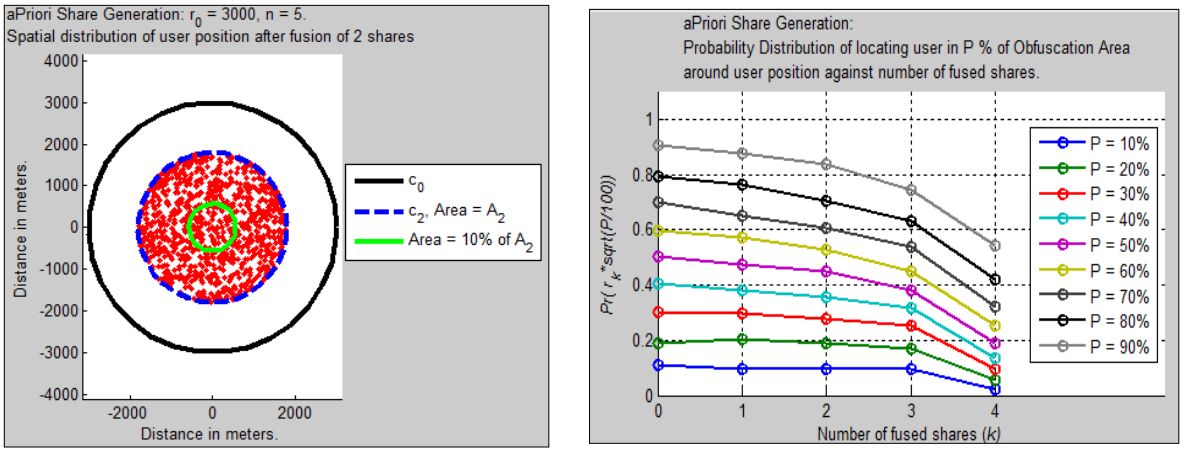


Figure 6.8. User position's distribution in *a-priori* share generation [3]: p_n(Red) in r_0 radius around p_0 using Monte Carlo simulation with n = 5 and 1000 runs.

However, the refinement shares now have limited choice of direction and length to connect p_0 and p_n. This results in inter-dependence of share vectors which implies that the more shares an attacker has, the more predictable p_n becomes through $P_{attack}(\phi)$ [3]. This is illustrated in Figure 6.5.

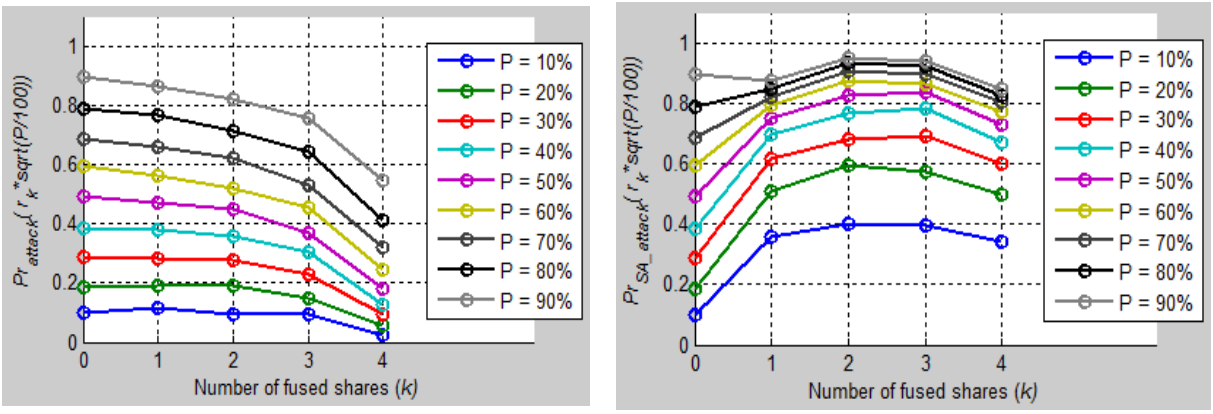


(a) (b)

Figure 6.9. Privacy security of a-priori implementation using 1000 runs of Monte Carlo simulation. a) Close to uniform distribution of p_n around p_2 in 10% area around p_2. b) $P_{attack}(\phi)$ for fusion of k shares.

Our implementation of *a-priori* share generation was different from the original algorithm of [3] as explained in sub-section 6.1.1. The $P_{attack}(\phi)$ for our implementation with $n = 5$ is shown in Figure 6.9. Part (a) of the figure illustrates the distribution of p_n in the obfuscation region c_2. It can be seen that the distribution throughout the obfuscation region is quite uniform as confirmed by part (b) of the figure for $k = 2$. When k is increased, distribution of user position becomes more and more non-uniform, especially if higher percentages of obfuscation area are considered. The decrease in probability of attack with increase of k means that the p_n start getting concentrated around the boundaries of the obfuscation region. Generally however, this implementation gives much closer results to uniform distribution than results from [3] shown in Figure 6.5.

Once again however, we need to see the effect of $P_{SA-attack}(\phi)$ and $P_{combined\ attack}(\phi)$ before judging the privacy protection of this approach. For the purpose of comparison, both, $P_{attack(\phi)}$ and $P_{SA-attack(\phi)}$, are plotted side by side in Figure 6.10.



(a) (b)

Figure 6.10. *a-priori* share generation for n= 5. a) $P_{attack}\ (\phi)$. b) $P_{SA-attack}\ (\phi)$.

By taking the best result from both of these attacks, the attacker can establish an even better estimate of precise user position. Best result here means selection of the probability value from the two probabilities, which represents a more non-uniform distribution than the other one. For example, the attacker can infer that the probability of finding the user in 10% of the obfuscation region after fusing 3 shares is 40%, from Figure 6.10 (b), rather than 10% (uniform) given by part (a) of the figure. Similarly, after fusing 4 shares, the attacker may also infer that in 10% of the obfuscation area, the chances of finding precise user location are approximately 2.5% from part (a) of the figure instead of approximately 37% from part (b). In this case, the fractional distance of 37% from the uniform distribution value 10% in the range $[10-100]\%$ is less compared to the fractional distance of 2.5% from 10% in the range $[0-10]\%$.

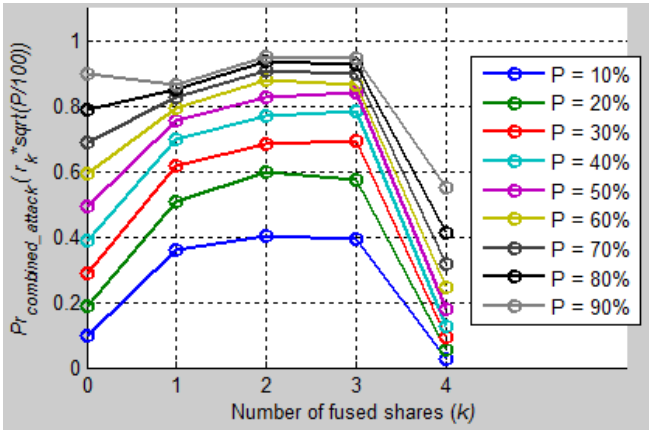


Figure 6.11. $P_{combined\ \ attack}\ (\phi)$ for *a-priori* share generation

By applying this principle to the results of both attacks, the attacker obtains a new and more informative distribution, $P_{combined\ attack}(\phi)$, about user position given in Figure 6.11.

6.3 Selection of GPS Traces

Real world GPS traces are ideal as test inputs for quantifying the performance achieved by DR, SU and SUaDR algorithms. For this purpose, we have downloaded traces from [24] & [25]. Other than separation of the traces into four main categories, no particular considerations were made during their search. These categories are:

1. Car traces on a long route
2. Car traces in urban area
3. Walk traces in unstructured environment
4. Walk traces in urban area.

These categorizations are important to see the viability of the algorithms in real life usage. These categories offer different ranges of speeds (see Table 6.1) and different turning angles. For example, turns in (ii) and (iv) are dominantly at right angles whereas in (i) and (iii), turn angles are less sharp.

Trace Set	Number of GPS Traces	Average Speeds(km/h)
Car traces on a long route	4	85.17
Car traces in urban area	4	24.77
Walk traces in unstructured environment	3	5.14
Walk traces in urban area	7	6.36

Table 6.1. Average speeds of the four GPS trace sets.

Due to the differences in trace sets, the values of parameters, as given in Table 6.2, were selected for the evaluation of SU, DR and SUaDR. The value of n is kept constant for all trace sets for easy comparison of their performance. However, r_0 was changed so that a reasonable amount of share generations are performed during the course of distance covered by a particular trace. As the distances travelled in traces, apart from car traces on long routes, were not very long, a comparatively smaller value of r_0 were chosen.

Trace Set	n	r_0 (m)	DR_Th (m)	SU_Th (m)
Car traces on a long route	5	3000	40	3000
Car traces in urban area	5	1000	30	1000
Walk traces in unstructured environment	5	500	15	500
Walk traces in urban area	5	500	15	500

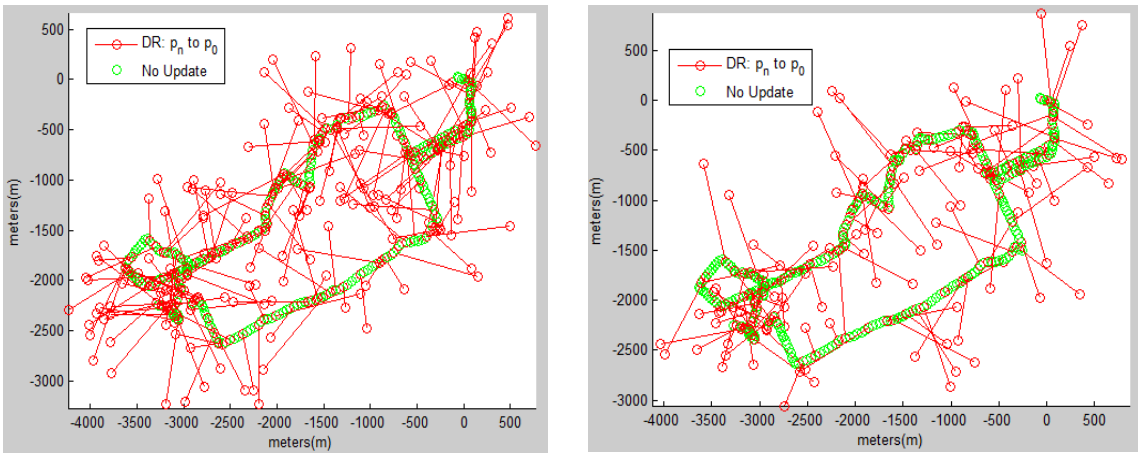
Table 6.2. Parameter values for testing with different GPS trace sets.

6.4 Individual Evaluation

At first, we will consider DR, SU and SUaDR in detail with regards to the achieved reduction in communication cost and their effect on the security of share generation algorithm i.e. $P_{attack}(\phi), P_{SA-attack}(\phi)$ and $P_{combined\ attack}(\phi)$. Then we will compare the results of application of all algorithms on the defined GPS trace sets. For individual evaluations, only *a-priori* share generation is considered. In comparative evaluations, both types of share generations will be tested.

6.4.1 DR

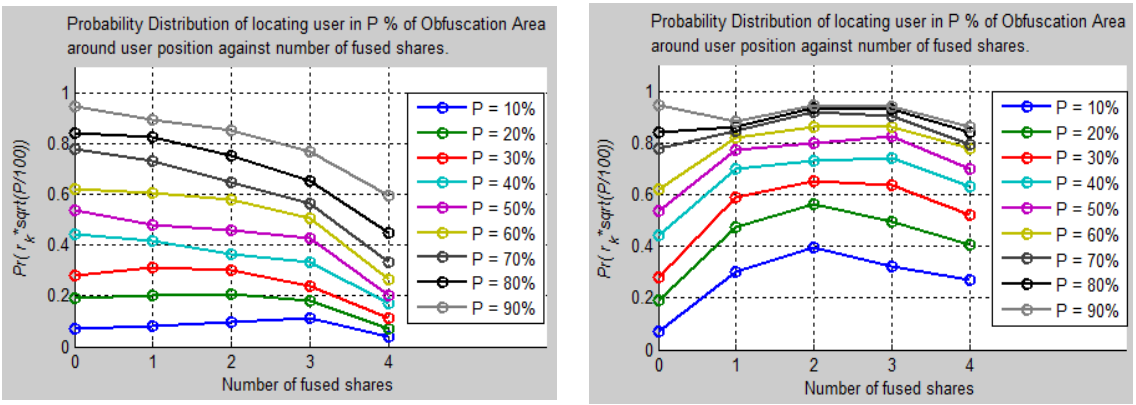
As mentioned earlier, the number of messages sent to the LSs using DR are directly dependent on the value of DR_Th. This is illustrated in Figure 6.12 in which DR is applied using *a-priori* share generation on a GPS trace. A clear difference is seen in run (a) and (b) in the number of complete share generations. Part (b) of the figure clearly shows less dense share generations (red lines) compared to part (a) due to doubling of the value of DR_Th.



(a) (b)

Figure 6.12. Impact of DR_Th on 359 position fixes of a urban car trace: $n = 5, r_0 = 1000$. a) $DR\_Th = 20$, 161 share generations, $PMR = 55\%$. b) $DR\_Th = 40$, 107 share generations, $PMR = 70\%$.

With regards to the influence of DR on privacy security, the results are quite similar to the original results for *a-priori* algorithm given in Figure 6.10 and Figure 6.11. They are presented in Figure 6.13 and Figure 6.14.



(a) (b)

Figure 6.13. DR results for 359 GPS fixes of a urban car trace. a) $P_{attack}(\phi)$. b) $P_{SA-attack}(\phi)$.

Generally, it was observed that DR does not affect the original security provided by the share generation algorithm. This is because the refinement shares generated by the share generation algorithm are not altered in any way. The re-usage of same shares, especially over straight patches of the trace, causes some difference to appear in the distribution of user position measured by

$P_{attack}(\phi), P_{SA-attack}(\phi)$ and $P_{combined\ attack}(\phi)$. However, these differences are not very significant.

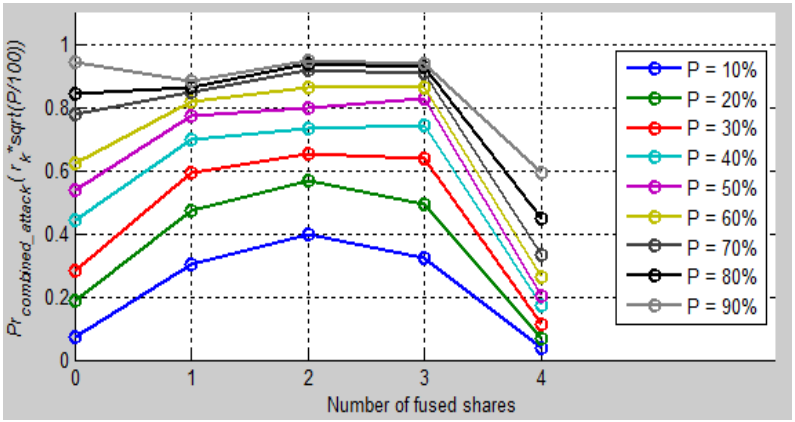
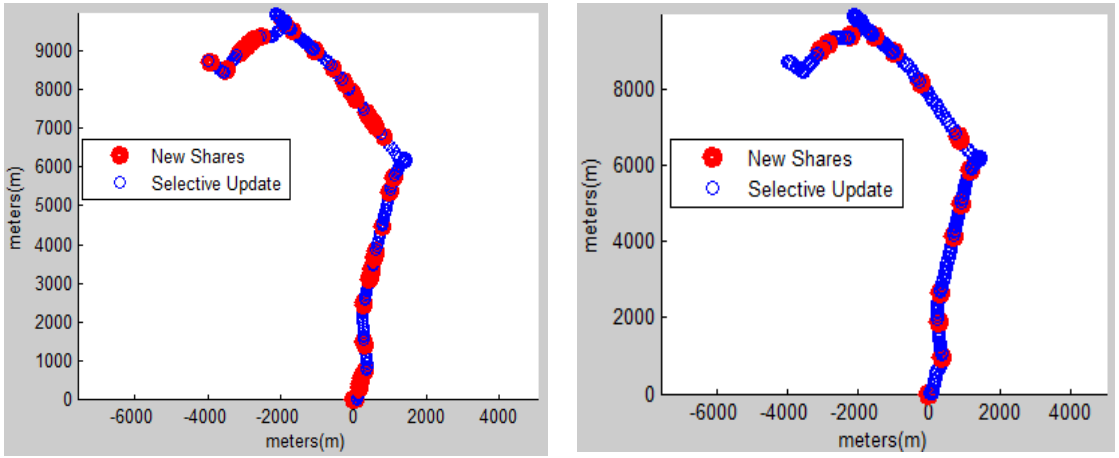


Figure 6.14. DR results: $P_{combined\ \ attack}\ (\phi)$ generated from Figure 6.13 (a) and (b).

For the test categories of GPS traces, the selected values of DR_Th are given in Table 6.2. The choice of these values was determined by experimentation in order to get good communication overhead reduction. The values are appropriate considering the speed of the user in each scenario.

6.4.2 SU

Similar to DR, performance of SU is also effected by SU_Th. Smaller value of SU_Th means that more share generations are needed as shown in Figure 6.15. The overall reduction in messages is seen to increase when SU_Th is doubled. It is important to note that the doubled SU_Th did not significantly increase the PMR i.e. only 8%.

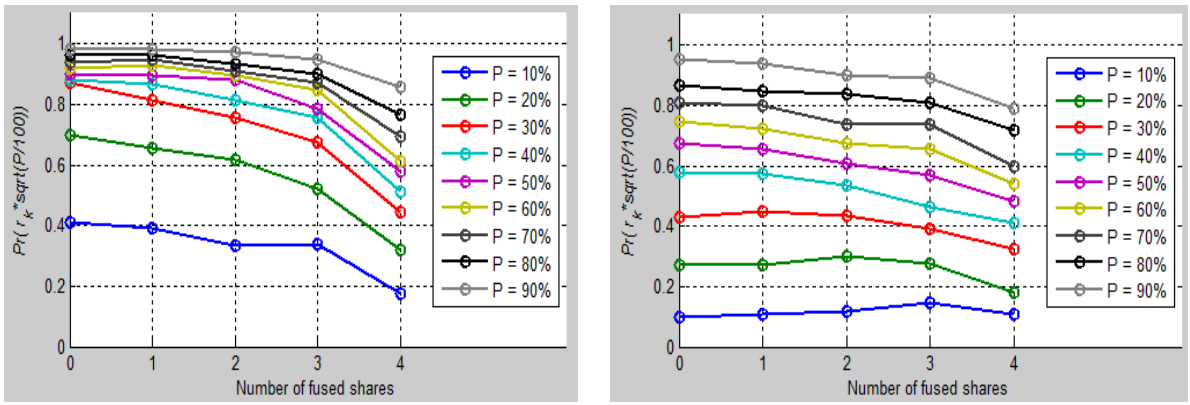


a) b)

Figure 6.15. SU run on 318 position fixes. *a-priori Share* generation with $n = 5, r_0 = 1000m$. a) $SU\_Th = 500m$: 64 new sets of shares generated, $PMR = 64\%$. b) $SU\_Th = 1000m$: 15 new sets of shares generated, $PMR = 72\%$.

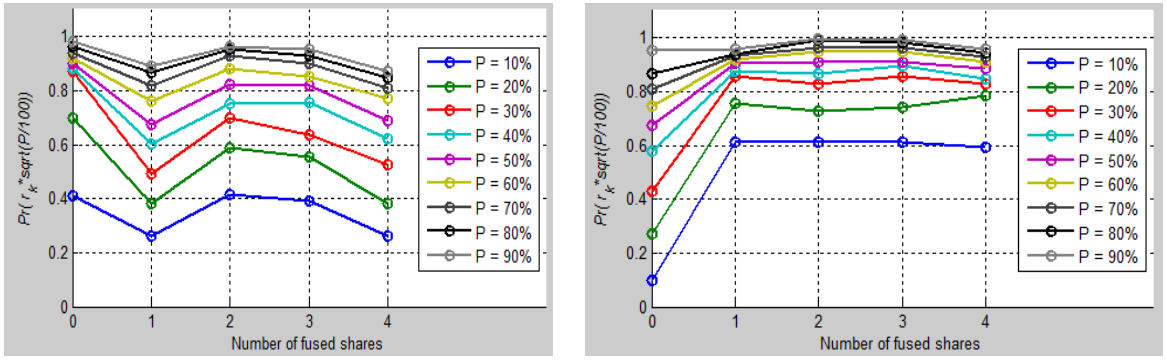
For quantifying the effect on security, Figure 6.16 shows that $P_{attack}(\phi)$ becomes more close to uniform when higher value of SU_Th is used. However, $P_{SA-attack}(\phi)$ is adversely affected by increase in SU_Th (see Figure 6.17). The precise position of user becomes more and more focused near the center of the

obfuscation region because the shares become stretched to their full length, thereby helping the SA attack. An overall decrease in uniformity of user's position distribution by the combination of the two attacks is seen as shown in Figure 6.18.



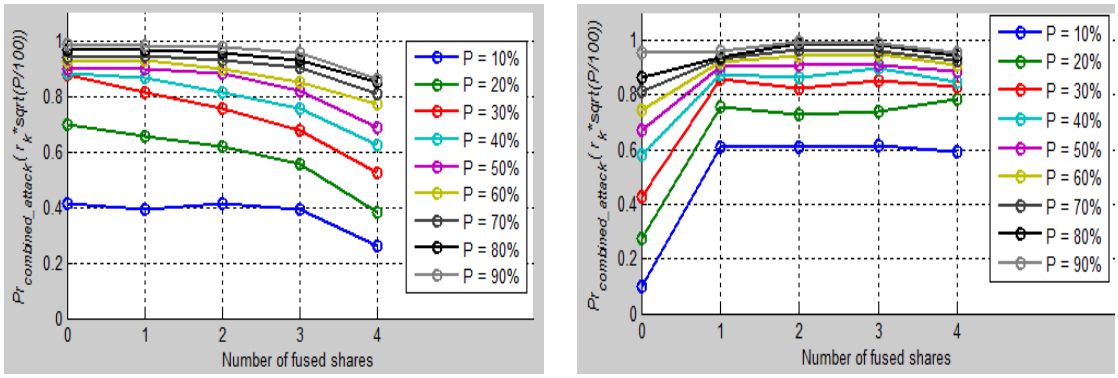
a) b)

Figure 6.16. $P_{attack}\ (\phi)$ for the SU applied GPS trace of 318 position fixes, *a-priori* share generation, $n = 5, r_0 = 1000m$. a) $SU\_Th = 500m$. b) $SU\_Th = 1000m$.



a) b)

Figure 6.17. $P_{SA-attack}\ (\phi)$ for the SU applied GPS trace of 318 position fixes, *a-priori* share generation, $n = 5, r_0 = 1000m$. a) $SU\_Th = 500m$. b) $SU\_Th = 1000m$.



a) b)

Figure 6.18. $P_{combined\ attack}\ (\phi)$ for SU applied GPS trace of 318 position fixes, $r_0 = 1000m$. a) $SU\_Th = 500m$. b) $SU\_Th = 1000m$.

6.4.3 SUaDR

In our evaluations, we have kept the value SU_Th used in SUaDR equal to r_0 in order to clearly see its effect. Figure 6.19 shows the application of SUaDR on a GPS trace using *a-priori* share generation. The resulting PMR of 79% is better than 75% achieved through DR and 72% by SU.

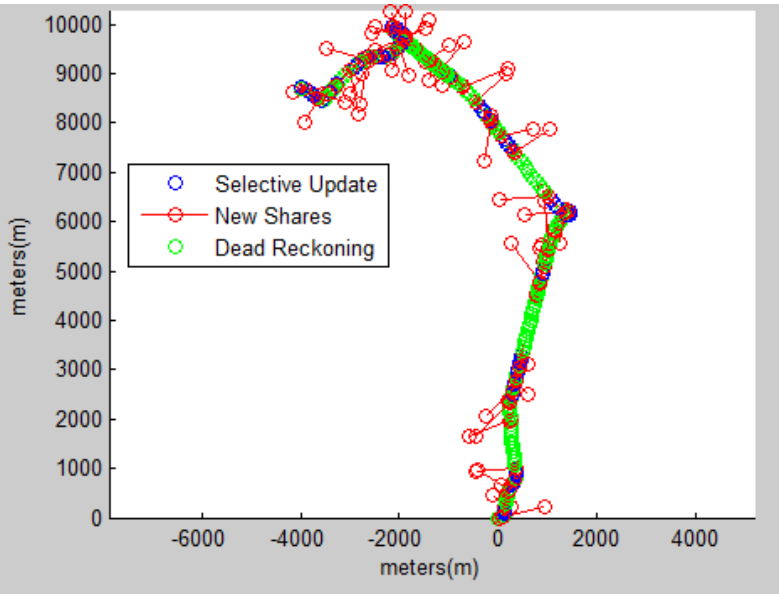


Figure 6.19. SUaDR applied to GPS trace with 318 position fixes. Acheived $PMR = 79\%$. Total 50 new share sets generated.

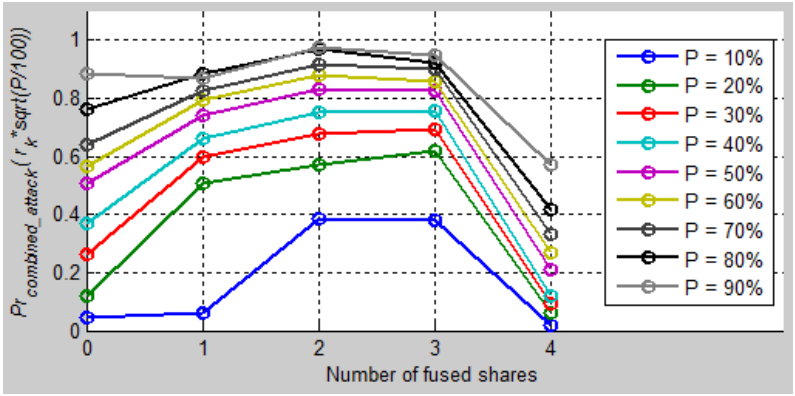


Figure 6.20. $P_{combined\ \ attack}\ (\phi)$ for SUaDR applied on GPS trace of Figure 6.19,.

Due to the reason that a considerable number i.e. 50, sets of new share were generated, the privacy security plot in Figure 6.20 is similar to that of *a-priori* share generation.

It is not always the case that SUaDR performs better compared to SU and DR with regards to the attained PMR. Another run, see Figure 6.21, on a different GPS trace yields a PMR of 57% which is higher than PMR of DR i.e. 38%, whereas less than that of SU (66%) for the same GPS trace. With poor performance of DR, SUaDR incorporates selective updates to reduce the effect of jerky trajectory and therefore attain high PMR compared to DR. Again, a

$P_{comibned\_attack}(\phi)$ similar to that of the previous trace is observed in Figure 6.22.

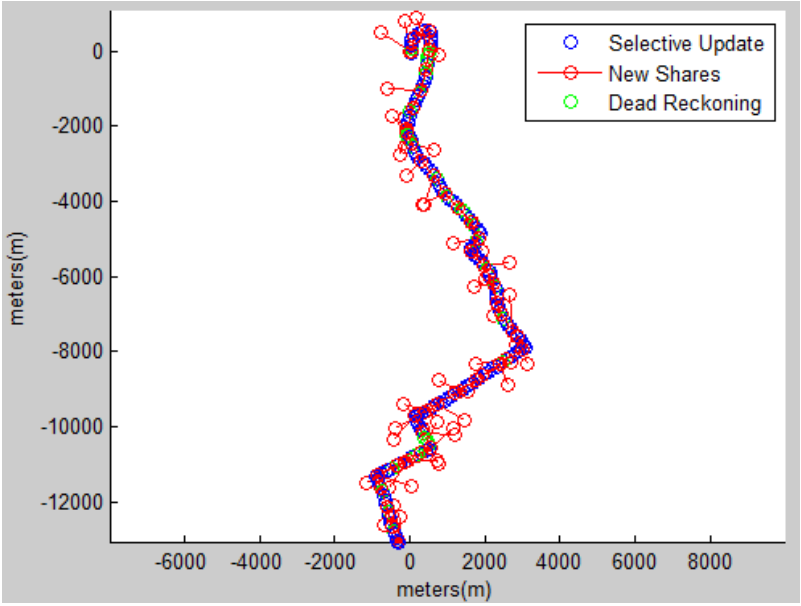


Figure 6.21. SUaDR applied to a GPS trace using *a-priori* share generation. Attained $PMR = 57\%$. New sets of shares generated $= 46$.

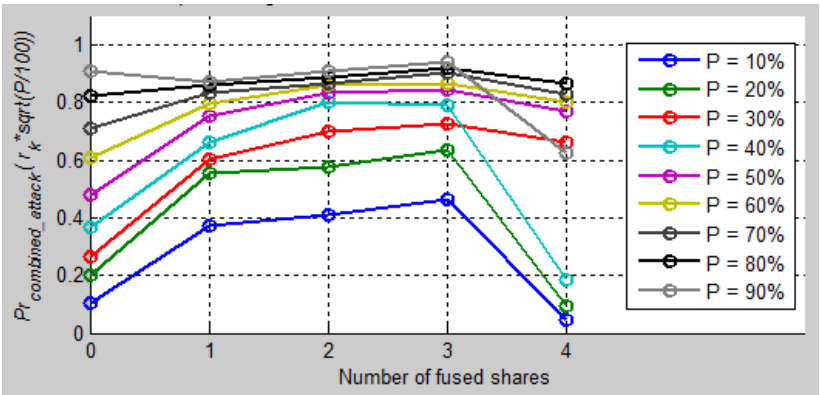


Figure 6.22. $P_{combined\ \ attack}\ (\phi)$ for SUaDR applied on GPS trace of Figure 6.21.

6.5 Comparative Evaluation

6.5.1 Communication Overhead Optimization

The summary of attained average PMR and $MPPF$ after application of the algorithms on the trace sets using algorithm parameter values from Table 6.2 are given in tables Table 6.3 and Table 6.4 for *a-priori* and in Table 6.5 and Table 6.6 for *a-posteriori* share generation. These results show that the SUaDR algorithm performs best with highest average value of PMR and lowest average value for $MPPF$ on all trace sets for both share generation algorithms.

Trace Set	SU	DR	SUaDR
Car traces on a long route	77.3	94.3	94.3
Car traces in urban area	72.0	61.8	69.6
Walk traces in unstructured environment	69.8	67.1	72.7
Walk traces in urban area	72.7	71.4	76.2
Average *PMRs*	72.95	73.64	78.21

Table 6.3. *a-priori* share generation: Average PMR of the GPS trace sets for communication optimization algorithms.

Trace Set	SU	DR	SUaDR
Car traces on a long route	1.13	0.29	0.28
Car traces in urban area	1.40	1.90	1.52
Walk traces in unstructured environment	1.50	1.64	1.37
Walk traces in urban area	1.37	1.43	1.19
Average *MPPFs*	1.35	1.32	1.09

Table 6.4. *a-priori* share generation: Average $MPPF$ of the GPS trace sets for communication optimization algorithms. Note that here $n = 5$.

Trace Set	SU	DR	SUaDR
Car traces on a long route	77.5	94.3	94.4
Car traces in urban area	71.3	61.8	72.0
Walk traces in unstructured environment	70.7	67.1	74.6
Walk traces in urban area	72.9	71.4	77.9
Average *PMRs*	73.1	73.64	79.71

Table 6.5. *a-posteriori* share generation: Average PMR of the GPS trace sets for communication optimization algorithms.

Trace Set	SU	DR	SUaDR
Car traces on a long route	1.13	0.29	0.28
Car traces in urban area	1.43	1.90	1.40
Walk traces in unstructured environment	1.47	1.64	1.27
Walk traces in urban area	1.35	1.43	1.10
Average *MPPFs*	1.34	1.32	1.01

Table 6.6. *a-posteriori* share generation: Average $MPPF$ of the GPS trace sets for communication optimization algorithms. Note that here $n = 5$.

Trace Set	Avg. Distance between position fixes (m)	Avg. Angle between position fixes (degrees)	Avg. Std of Angle between position fixes (degrees)
Car traces on a long route	33.3	1.42	5.7
Car traces in urban area	55.8	12.4	21.4
Walk traces in unstructured environment	30.7	14.6	17.8
Walk traces in urban area	24.7	22.5	29.8

Table 6.7. Characteristics of GPS trace sets

The performance of DR is not effected by the share generation algorithm because the production of a new set of shares is independent of the share generation policy. DR performs better with long route car traces because high speeds make sharp turns difficult, thus increasing the predictability of user position. To quantify this effect, Table 6.7 shows that average angle between consecutive position fixes for long route car traces was 1.42 degrees which was the least of all trace sets. Also the position fix density represented by average distance between position fixes for long route car traces was less considering the average speed of the user on the track. On urban area car traces, the less density of position fixes ($55.8m$) along with high average change of direction(12.4 degrees) can be a reason for low attained value of $PMR(61.8\%)$. For the walk trace sets, DR shows reasonable performance considering the trace characteristics.

The performance of SU is consistent in all sets of GPS traces with an average PMR of around 73%. This is mainly because the length of a single share in each scenario was big enough to accommodate the average distance between traces and variations in movement direction of the user. If this was not the case, then more share updates than one would be needed, on average, to make the LSs current for each new position fix. The overall performance was again not effected significantly by a change in share generation algorithm.

SUaDR performs considerably well with PMR above 78% for both share generation algorithms. It exploits the consistent performance of SU in traces where DR does not work. Therefore, it improves the best of the individual SU and DR algorithms to attain an overall highest average PMR for all trace sets and both types of share generations.

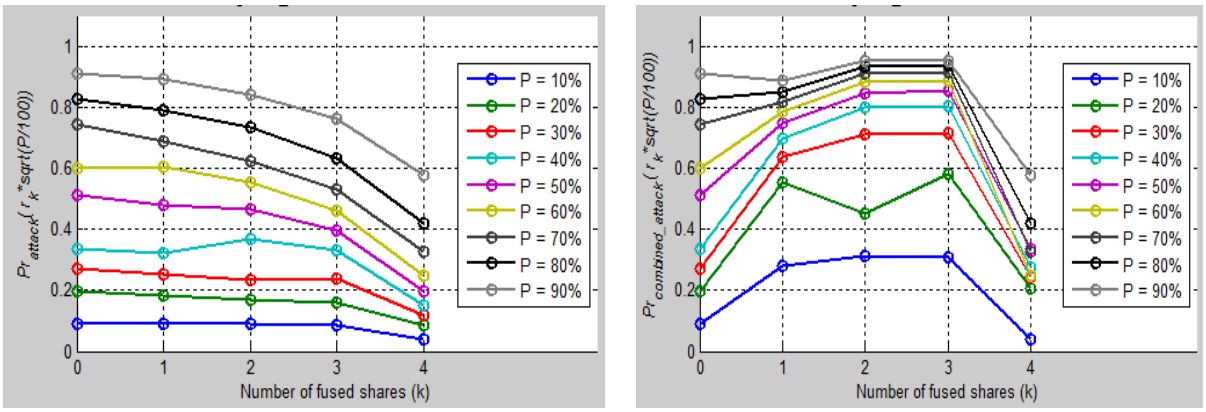
With regards to $MPPF$, we should note that it is a relative term as a comparison with non-privacy aware non-optimized systems where there is one location update message sent per position fix. By increasing the value of n, we can get an understanding about the practical feasibility of our algorithms. With $n = 5$, Table 6.4 gives an average $MPPF$ of 1.1 for SUaDR. However, when retested with $n = 8$, SUaDR gave an $MPPF$ of 1.53 . This sharp rise in its value dictates that the value of n should be carefully chosen considering the cost of communication in spite of overhead optimization.

6.5.2 Security Analysis

To summarize the effect of the algorithms on privacy security, we have only considered $P_{attack}(\phi)$ and $P_{combined\ attack}(\phi)$ because $P_{combined\ attack}(\phi)$ already includes the effect of $P_{SA-attack}(\phi)$. The following results are for *a-priori* share generation. In the end of sub-section 6.5.2, we will also present the average results on all tracks for *a-posteriori* share generation.

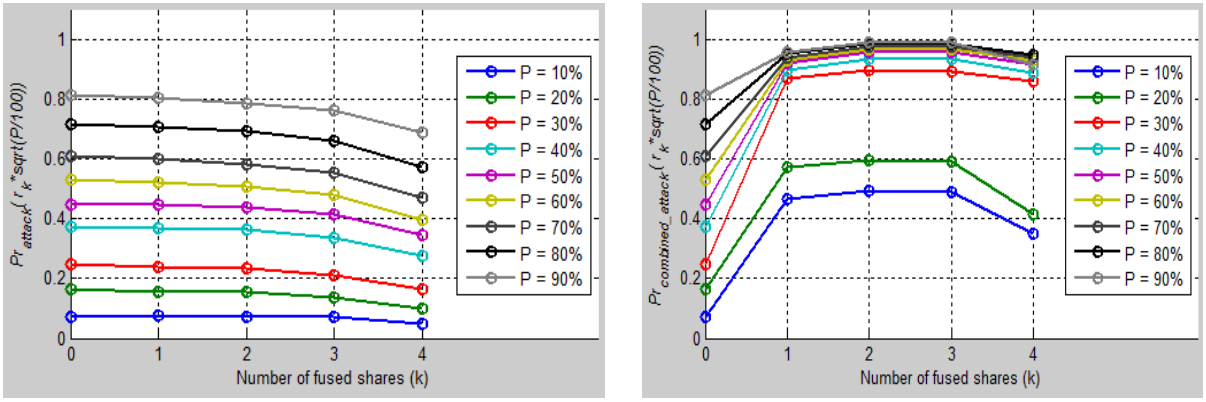
6.5.2.1 Car traces on long route

The results for SUaDR, SU and DR can be seen in Figure 6.23, Figure 6.24 and Figure 6.25. For SUaDR and DR, both $P_{attack}(\phi)$ and $P_{combined\ attack}(\phi)$ are very similar to that of the original *a-priori* algorithms. However, SUaDR reveals less information about precise user location when considering only 10% of the obfuscation area. For SU, the uniformity of $P_{attack}(\phi)$ does increase but $P_{combined\ attack}(\phi)$ shows introduction of considerable non-uniformity due to SA attack when 30% or more of the obfuscation area are considered.



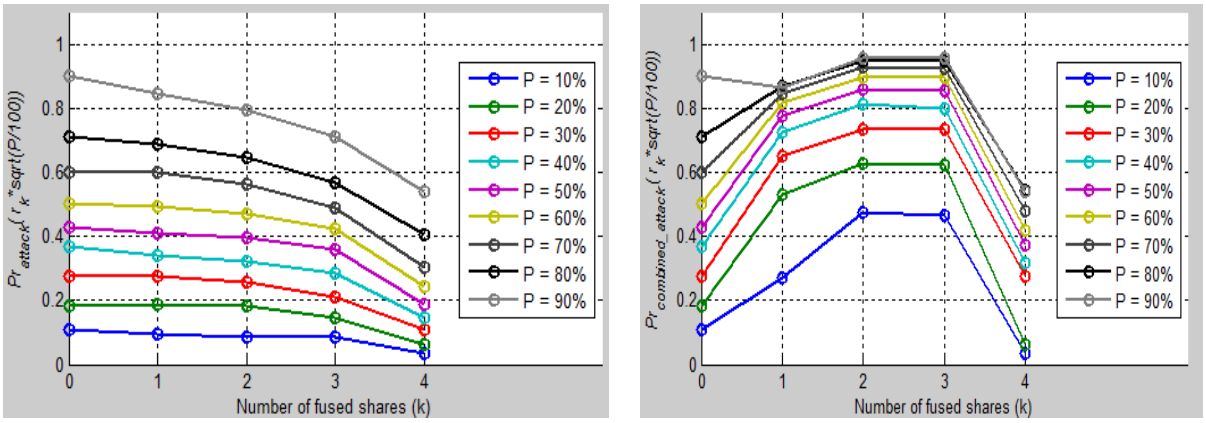
a) b)

Figure 6.23. SUaDR averages for long route car traces: a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.



a) b)

Figure 6.24. SU averages for long route car traces: a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.

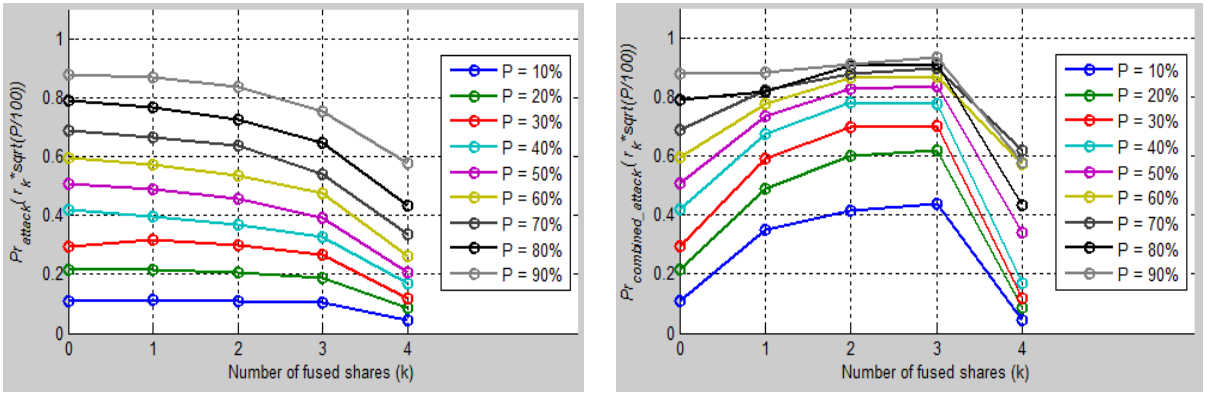


a) b)

Figure 6.25. DR averages for long route car traces: a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.

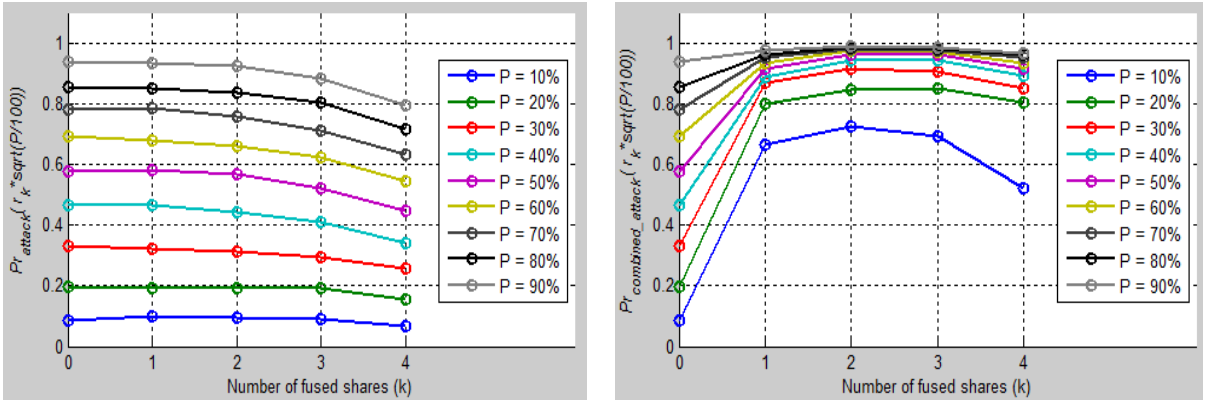
6.5.2.2 Car traces in urban area

Again, SU is outperformed by SUaDR and DR when $P_{combined\ attack}(\phi)$ is considered. DR performs slightly better than SUaDR when 10% of the obfuscation area is considered.



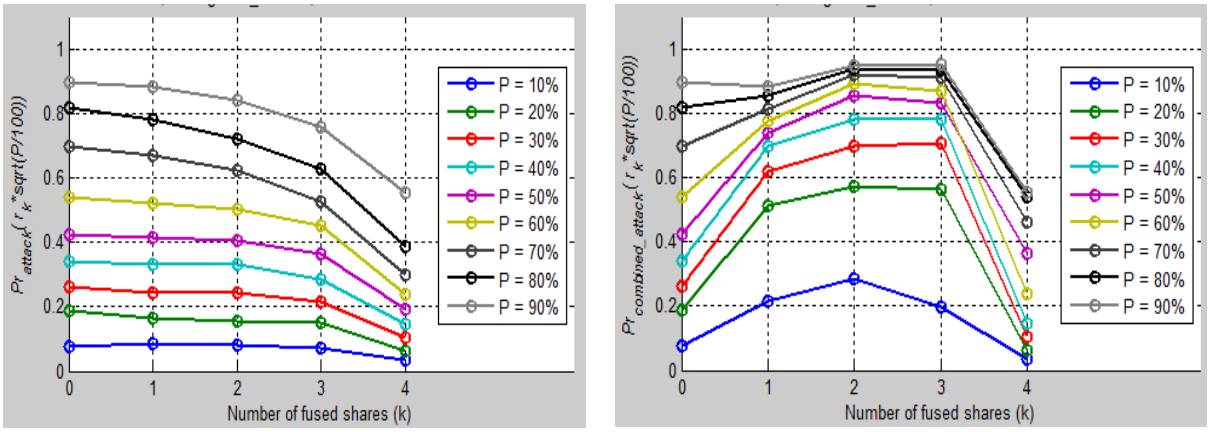
a) b)

Figure 6.26. SUaDR averages for urban area car traces: a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.



a) b)

Figure 6.27. SU averages for urban area car traces: a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.

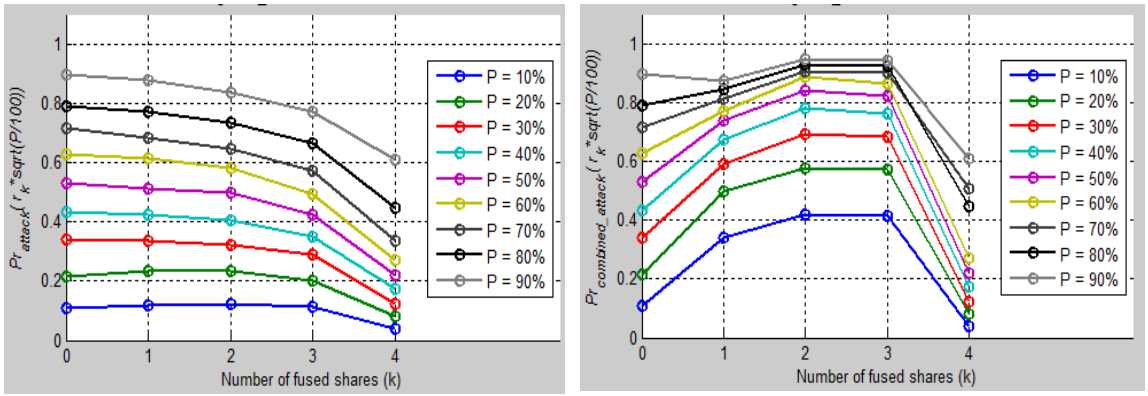


a) b)

Figure 6.28. DR averages for urban area car traces: a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.

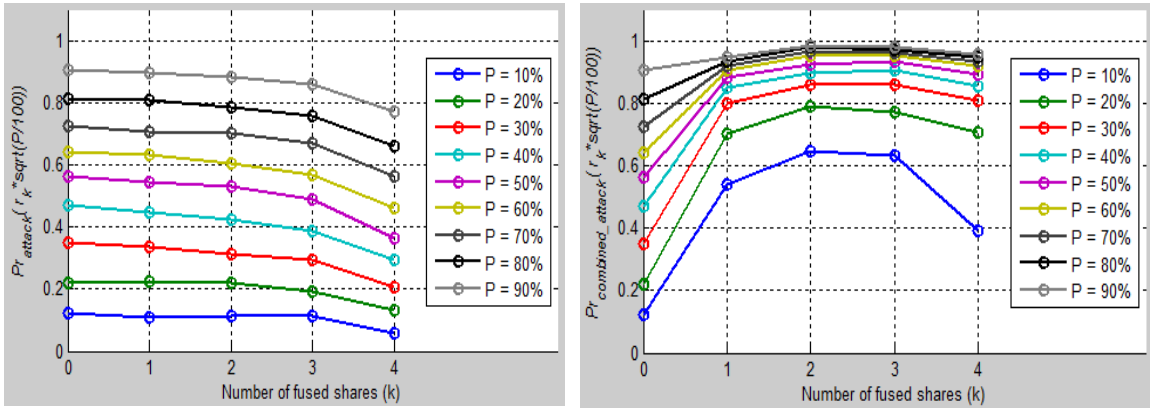
6.5.2.3 Walk traces in unstructured environment

Here again we see the same type of results. DR provides the best security by improving the security provided by the *a-priori* share generation.



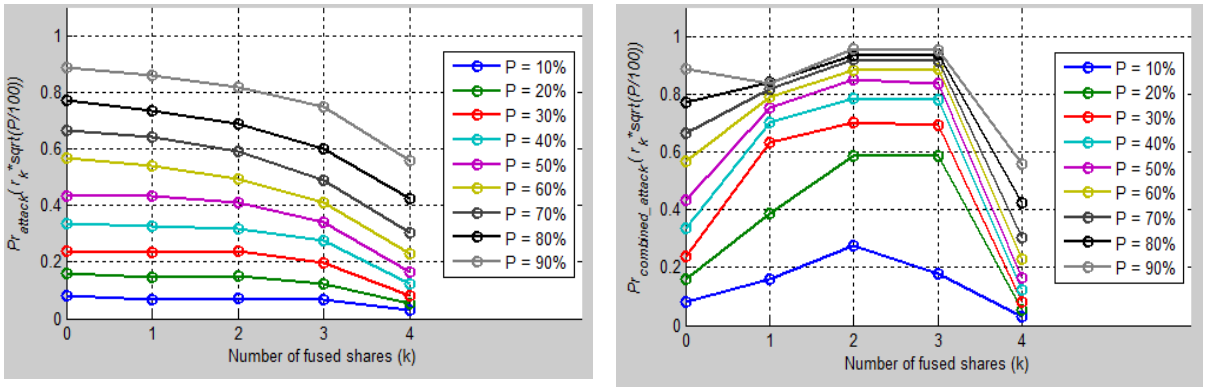
a) b)

Figure 6.29. SUaDR averages in unstructured area walk traces : a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.



a) b)

Figure 6.30. SU averages in unstructured area walk traces : a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.

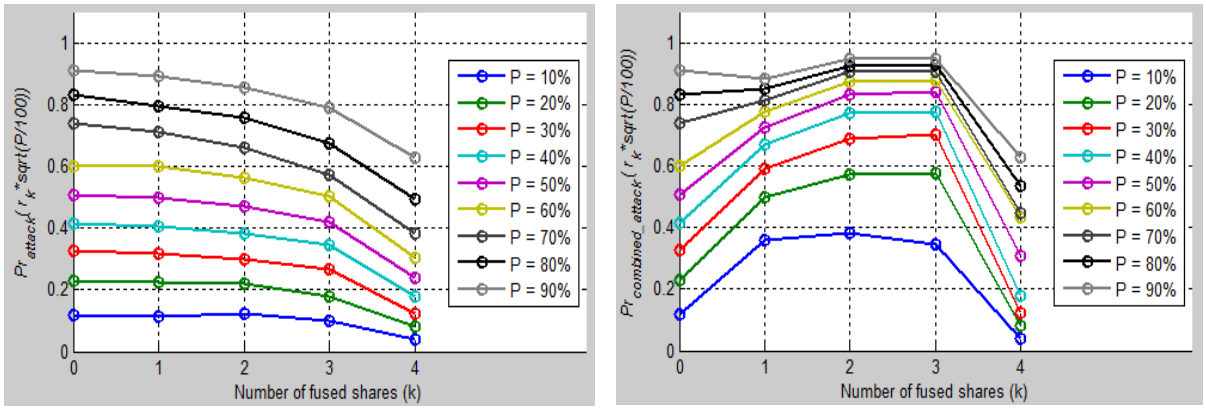


a) b)

Figure 6.31. DR averages in unstructured area walk traces : a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$

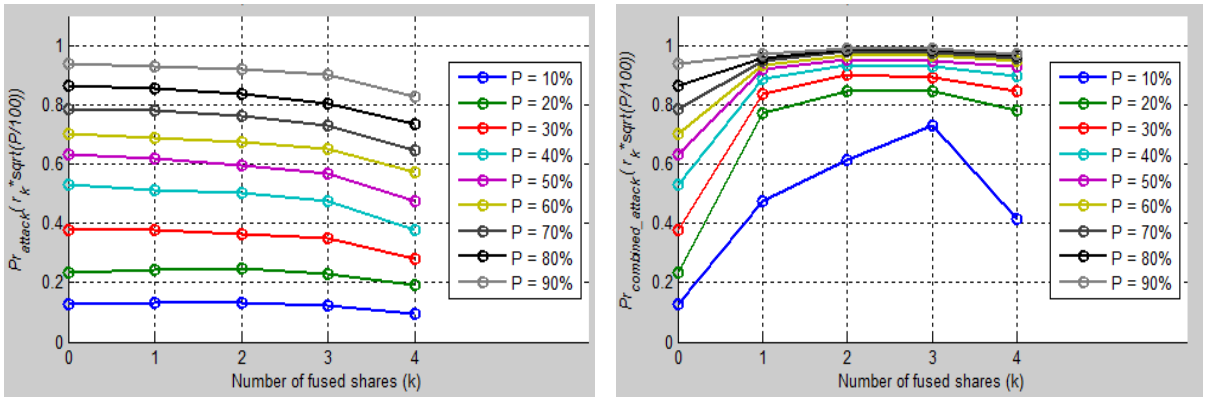
6.5.2.4 Walk traces in urban area

SUaDR and DR again maintain the privacy security of the original *a-priori* share generation. SU again suffers due to the SA attack.



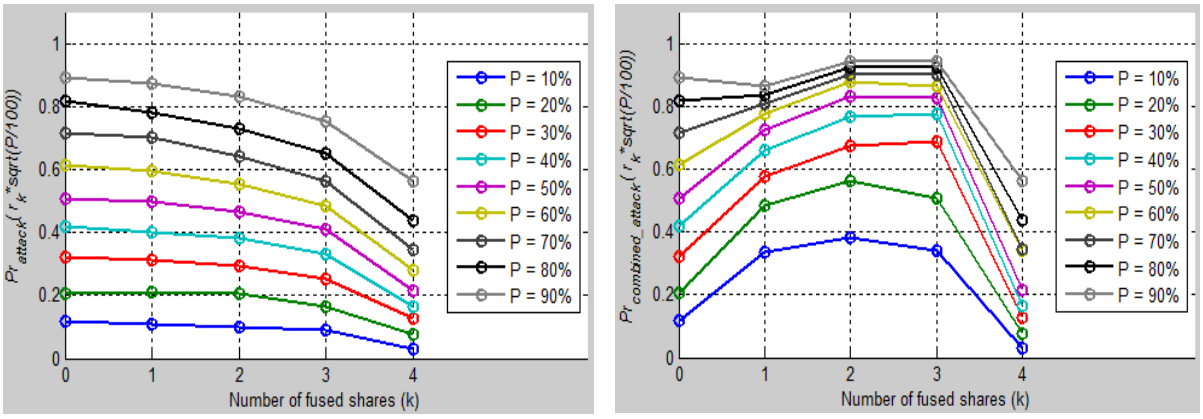
a) b)

Figure 6.32. SUaDR averages in urban area walk traces : a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.



a) b)

Figure 6.33. SU averages in urban area walk traces : a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.



a) b)

Figure 6.34. DR averages in urban area walk traces : a) $P_{attack}\ (\phi)$. b) $P_{combined\ \ attack}\ (\phi)$.

6.5.2.5 *Results with a-posteriori* share generation

For *a-posteriori* share generation, we have taken the average of $P_{combined\ attack}(\phi)$ on all trace sets. The general behavior of each algorithm is captured well by the figures below. It is noticeable that SU again changes the behavior of the original algorithm (see Figure 6.35). Generally, with fusion of increasing number of shares, the uniformity of user position in the obfuscation area decreases until the fourth share is fused. This behavior is different from the behavior of original *a-posteriori* share generation algorithm depicted in Figure 6.7.

On the other hand, DR and SUaDR almost maintain the security characteristics of the original algorithm as shown in Figure 6.35 and Figure 6.36 respectively.

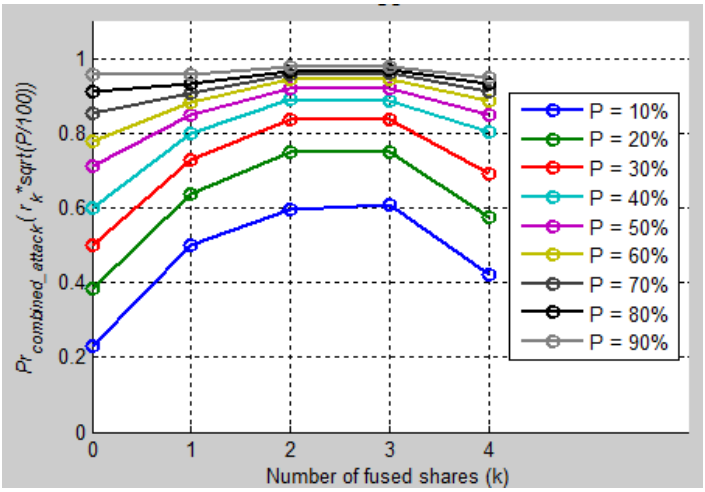


Figure 6.35. SU effect on *a-posteriori* share generation: $P_{combined\ \ attack}\ (\phi)$

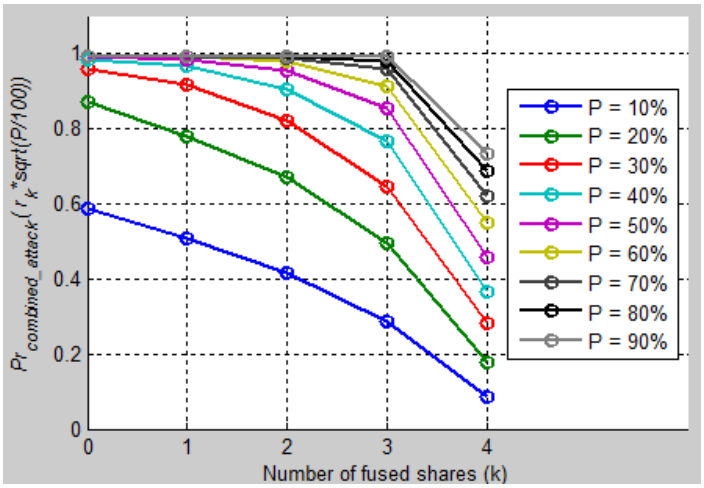


Figure 6.36. DR effect on *a-posteriori* share generation: $P_{combined\ \ attack}\ (\phi)$

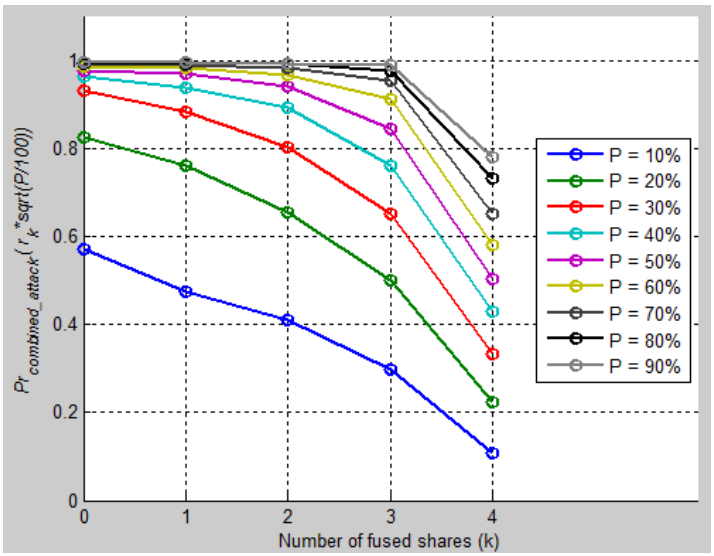


Figure 6.37. SUaDR effect on *a-posteriori* share generation: $P_{combined\ \ attack}\ (\phi)$

Considering the communication overhead reduction and the provided privacy security, SUaDR can be called the best algorithm with highest value of PMR and almost no effect on the privacy security provided by the share generation algorithm. By utilizing the good security traits of DR and adaptability of SU, SUaDR provides an efficient and consistent communication optimization solution for PS.

7. Summary and Future Work

The problem of discretely sharing location data presents a challenging problem, especially in non-trusted system. Position Sharing (PS) [3] presents a solution to this issue with generation of significant communication overhead. In this thesis, an attempt was made to reduce this communication overhead while maintaining the privacy guarantees of the PS algorithm. To this end, different overhead optimization schemes including those employed in non-privacy aware systems were formulated for PS and tested as possible solutions. These included the Dead Reckoning (DR) and the Selective Update (SU). From observation of their behavior, a third technique called SUaDR, which merged the two, was also proposed. The result of evaluations of the techniques on categorized sets of real world GPS traces has shown that significant optimization can be achieved by DR, SU and SUaDR in long route and urban car traces as well as walk traces. It was also shown that the privacy guarantees of the original PS algorithm were also maintained by the DR and SUaDR schemes.

During the evaluation part, it was discovered that a Share Average (SA) attack can reveal further information about user position. This attack has also adversely affected the privacy guarantees of original share generation algorithms. As part of future work, research into reducing the effectiveness of SA attack can be attempted. This work mainly focused on reducing the communication overhead of the original PS algorithm by reduction of messages instead of message size. Therefore, an attacker could possibly judge the span of user movements by the number of message sent to the LSs, especially in the case of SU. For protection against these and multiple position attacks, such as maximum movement boundary attack, further improvement and testing of the overhead optimization techniques may be explored.

8. Bibliography

[1] P. S. F. D. K. R. Marius Wernke, "A Classification of Location Privacy Attacks and Approaches," *Pervasive and Mobile Computing,* pp. 1-13, 2013.

[2] "Chronology of Data Breaches," Privacy Rights Clearinghouse, 19 November 2012. [Online]. Available: http://www.privacyrights.org/data-breach.

[3] P. S. K. R. Frank Dürr, "Position Sharing for Location Privacy in Non-trusted Systems," in *9th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 128-135.*, Seattle, USA, 2011.

[4] "Tell-all telephone| Data Protection | Digital | ZEIT ONLINE," [Online]. Available: http://www.zeit.de/datenschutz/malte-spitz-data-retention.

[5] J. Krumm, "Inference attacks on location tracks," in *Proceedings of the 5th International Conference on Pervasive Computing (Pervasive '07), pp. 127{143Proceedings of the 5th Inter-national Conference on Pervasive Computing (Pervasive '07), pp. 127-143*, Toronto, Canada, 2007.

[6] G. G. K. M. D. P. Panos Kalnis, "Preventing Location-Based Identity Inference in Anonymous Spatial Queries," in *Knowledge and Data Engineering, IEEE Transactions on , vol.19, no.12, pp.1719,1733*, Dec. 2007.

[7] M. F. Mokbel, "Privacy in location-based services: State-of-the-art and research directions," in *International Conference on Mobile Data Management*, Mannheim, 2007.

[8] D.-I. C. N. P. D. K. R. Dipl.-Inf. A. Leonhardi, "A Map-based Dead-reckoning Protocol for Updating Location Information," Institute of Parallel and Distributed High-Performance Systems (IPVR), Stuttgart, 2001.

[9] A. P. S. S. C. Y. Y. Ouri Wolfson, "Updating and Querying Databases that Track Mobile Units," *Distributed and Parallel Databases,* vol. 7, no. 3, pp. 257-387, 1999.

[10] L. L. P. P. T. W. Bhuvan Bamba, "Supporting anonymous location queries in mobile environments with privacygrid," in *17th International conference on World Wide Web (WWW '08), pp 237-246*, New York, USA, 2008.

[11] T. L. V. S. Ninghui Li, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *23rd IEEE International Conference on Data Engineering, (ICDE '07), pp. 106-115*, Istanbul, April 2007.

[12] F. S. S. Josep Domingo-Ferrer, "Micro-aggregation-based heuristics for p-sensitive k-anonymity: one step beyond," in *International workshop on Privacy and Anonymity in Information Society (PAIS '08), pp 61-69*, New York, USA, 2008.

[13] F. S. Alastair R. Beresford, "Mix zones: user privacy in location-aware services," in *Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, PERCOMW*, March 2004.

[14] M. C. E. D. S. D. C. a. d. V. P. S. C.A. Ardagna, "Location Privacy Protection Through Obfuscation-Based Techniques," in *21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Redondo Beach, CA, USA, 2007.

[15] Y. Y. T. S. Hidetoshi Kido, "An anonymous communication technique using dummies for location-based services," in *International Conference on Pervasive Services, 2005. ICPS '05, pp 88-97*, July, 2005.

[16] V. G. L. I. Pravin Shanka, "Privately querying location-based services with SybilQuery," in *11th international conference on Ubiquitous computing, pp. 31-40*, New York, USA, 2009.

[17] A. Gutscher, "Coordinate transformation - a solution for the privacy problem of location based services?," in *20th International Conference on Parallel and Distributed Processing Symposium IPDPS, pp. pp. 354-354*, Rhodes Island, Greece, 2006.

[18] C. S. J. J. M. H. L. Man Lung Yiu, "Design and analysis of a ranking approach to private location-based services," *ACM Transactions on Database Systems (TODS), New York, USA,* vol. 36, no. 2, p. 42, 2011.

[19] S. I. A. Nilothpal Talukder, "Preventing multi-query attack in location-based services," in *Proceedings of the third ACM conference on Wireless network security (WiSec), pp. 25-36*, New Jersey, USA, 2010.

[20] M. L. D. C. S. E. B. Gabriel Ghinita, "Preventing velocity-based linkage attacks in location-aware applications," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 246-255*, Seattle, Washington, 2009.

[21] F. D. K. R. Pavel Skvortsov, "Map-aware Position Sharing for Location Privacy in Non-trusted Systems," in *J. Kay et al. (Eds.): Pervasive, LNCS 7319, pp. 388-405, Springer Berlin*, Heidelberg, 2012.

[22] B. S. Ulrich Steinhoff, "Dead reckoning from the pocket - An experimental study," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mannheim, 2010.

[23] "MathWorks - MATLAB and Simulink for Technical Computing," The MathWorks, Inc., [Online]. Available: http://www.mathworks.com/.

[24] "OpenStreetMap," [Online]. Available: http://www.openstreetmap.org/copyright. [Accessed 12 03 2013].

[25] "GPSLib - GPS tracks hosting service," [Online]. Available: http://www.gpslib.net/tracks/.

Declaration

All the work contained within this thesis except where other acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Zohaib Riaz)