The Computer Journal Advance Access published February 11, 2015

doi:10.1093/comjnl/bxv009

On Designing Resilient Location-Privacy Obfuscators

Pericle Perazzo[1,*], Pavel Skvortsov[2] and Gianluca Dini[1]

[1]*Department of Information Engineering, University of Pisa, Pisa, Italy*
[2]*Institute of Parallel and Distributed Systems (IPVS), University of Stuttgart, Stuttgart, Germany*
**Corresponding author: pericle.perazzo@iet.unipi.it*

The success of location-based services is growing together with the diffusion of GPS-equipped smart devices. As a consequence, privacy concerns are raising year by year. Location privacy is becoming a major interest in research and industry world, and many solutions have been proposed for it. One of the simplest and most flexible approaches is obfuscation, in which the precision of location data is artificially degraded before disclosing it. In this paper, we present an obfuscation approach capable of dealing with measurement imprecision, multiple levels of privacy, untrusted servers and adversarial knowledge of the map. We estimate its resistance against statistical-based deobfuscation attacks, and we improve it by means of three techniques, namely *extreme vectors*, *enlarge-and-scale* and *hybrid vectors*.

Keywords: privacy; location-based services; obfuscation techniques; uniform obfuscation

Received 29 April 2014; revised 10 October 2014
Handling editor: Kui Ren

1. INTRODUCTION

Smartphones and other mobile smart devices are nowadays commonly equipped with GPS receivers. Increasingly more, the users send their position to remote providers in order to obtain location-based services (LBSs). The privacy concerns stemming from this are not negligible. The position of the user is personal data, and its diffusion to untrusted entities can bring several risks, from annoying targeted spam to stalking. Moreover, more sensitive information can be inferred from a person's position: her private habits, her religious beliefs, her political affiliation and so on. As the public awareness about these problems grows, the LBSs of the future will increasingly focus on privacy.

In the last years, the research world has developed many solutions for guaranteeing location privacy to a certain extent [1–5]. Often, these approaches are quite complex, and they require substantial modifications on the existing LBSs. A simple but flexible approach is *obfuscation*, by which the positions are perturbed with a random noise, thus artificially degrading their precision before releasing them to untrusted entities. This approach does not involve third trusted entities, burdensome cryptographic functions or energy-expensive peer-to-peer communications. This makes it very attractive for resource-constrained devices. Obfuscation approach has recently gained the attention of industry [6].

Despite their conceptual simplicity, obfuscation algorithms must be carefully designed. Otherwise, they are vulnerable to deobfuscation attacks based on *statistical analysis*. In these attacks, the adversary tries to 'invert' the obfuscation by computing the spatial probability distribution of the real user's position. Often, such a distribution is highly concentrated in some areas, and this makes the user's position quite predictable. Past studies [3] have showed that this vulnerability is present in many state-of-the-art obfuscation algorithms.

In this paper, we present a location obfuscation system capable of dealing with measurement imprecision, multiple levels of privacy, untrusted servers and adversarial knowledge of the map. We study its resistance against deobfuscation attacks, and we improve it by means of three techniques, namely *extreme vectors*, *enlarge-and-scale* and *hybrid vectors*. Our tests show that extreme vectors can decrease the adversarial success probability by 22.50%, enlarge-and-scale technique by 31.74% and hybrid vectors by 17.17%. The present work is both an integration and a follow-up of our previous works [2, 3, 7]. Such works have been improved in terms of resistance by means of extreme vectors, enlarge-and-scale and hybrid vectors.

The rest of the paper is organized as follows. Section 2 presents significative-related works. Section 3 describes in detail our basic obfuscation system. Section 4 models our adversary and introduces a metric to evaluate the resistance against deobfuscation attacks. Sections 5–7 describe, respectively, extreme vectors, enlarge-and-scale and hybrid vectors techniques, and evaluate their relative resistance improvement on the basic obfuscation system. Finally, the paper is concluded in Section 8.

2. RELATED WORK

Location privacy is gaining attention not only in the scientific circles, but also on the level of social awareness. The easiest countermeasure against leaks and misuses of user's location information would be a total denial of usage of the corresponding services. However, the comfort provided by widespread applications such as Foursquare [8], Loopt [9] and Google Latitude [10] became very important in modern life. In most cases people are looking for a solution that provides for a trade-off between privacy and some service quality, e.g. precision.

Approaches for location privacy can be divided in *identity protection* and *data protection*. The aim of identity protection is to avoid the re-identification of anonymous users. The aim of data protection is to avoid the disclosure of their personal data.

k-anonymity [11] is the mainstream approach for identity protection. This approach requires that the identity of the user is indistinguishable with at least $k-1$ other identities. Gruteser and Grunwald [12] first introduced it for LBSs. Since this method does not permit the identification of the user, it is not applicable in services in which the user authenticates himself, e.g. location-based social networks. In addition, it requires a centralized anonymizer, whose presence is usually considered a major drawback. Disregarding how reliable the approach is, one cannot exclude that the anonymizer itself is malicious or unreliable.

Chow *et al.* [13] proposed a technique to reach k-anonymity without an anonymizer, but it requires burdensome peer-to-peer communication between mobile devices. Our approach does not aim at identity protection, but rather at data protection. As a consequence, it is employable also in LBSs in which the user authenticates himself. In addition, it does not require a central trusted party, nor peer-to-peer communications.

One of the simplest methods for data protection is to create fake positions (*dummies*) and send them together with the true user position [1]. This increases the processing overhead on the LBS side, since LBS needs to send back a query response for each of the given position. This method is not widely accepted since usually the fake locations are easily distinguishable from the true ones.

Another data-protection approach is in Mascetti *et al.* [14] that consider the scenario where a mobile user wants to notify about his proximity to his friends (called *buddies*) in term of untrusted service providers. The secret keys are shared not with service providers but only with the selected buddies in a decentralized way. The utilized precision metric is defined through the union of multiple *granules*, i.e. discrete space cells. In general, the approach requires more complex system implementation due to encryption functionalities—but it is set up only for the specific proximity calculation use case of the LBS.

Ghinita *et al.* [15] focus on *private information retrieval* techniques for location data protection. The aim is to deliver a LBS without disclosing the user's position at all. This approach offer high security, but involves complex cryptographic operations, which scale poorly at the server side.

The *obfuscation* approach for location data protection has been introduced by Ardagna *et al.* [4]. They proposed a set of *obfuscation operators* that perturb the location in various ways. They do not explore the possibility that the adversary performs a deobfuscation by means of statistical analysis. A recent example of location obfuscation is the *n-CD* approach of [5]. Authors proposed the generation of such concealed disks (CDs), whose combination gives an obfuscated user's position. The overlapping and random rotation of CDs preserves unpredictability of the resulting obfuscation area (called 'anonymity zone'). The CDs are selected in such a way that their overlapping is consistent. The n-CD approach does not guarantee multiple privacy levels. Moreover, the proposed privacy metric considers only the resulting intersection area of the CDs, without analyzing the probability distribution of the target position within the anonymity zone. Li *et al.* [16] presents a general attack against proximity-based social networks such as WeChat [17] and MoMo [18], which allows a malicious user to discover the real position of another user. The authors also proposed a countermeasure to that based on obfuscation. In this paper, we do not focus on proximity-based social networks, but on a broader range of applications. Our approach could be applied to protect proximity-based social networks as well.

Probabilistic location privacy metrics were introduced by Shokri *et al.* [19] Authors proposed a general framework based on Bayesian Stackelberg game theory, which estimates probabilistic privacy levels provided by existing location privacy approaches. Authors have tested this framework by applying it to a obfuscation technique that relies on space discretization. In our work, we use probabilistic metrics to measure the privacy levels provided by our position sharing approach; the probabilistic privacy levels are provided in a non-discrete way and they can be flexibly managed depending on who is accessing the mobile user's position.

The approach of uniform obfuscation was presented in [3]. The idea is to make the traditional obfuscation through a circular area by taking into account the positioning inaccuracy. The basic approach of position sharing was introduced in [2] and extended with a map-aware version in [7] by adjusting

the obfuscation area depending on the privacy-sensitivity of the neighboring map objects. This paper presents both an integration and an improvement of [2, 3, 7] approaches that will be presented later in detail.

3. BASIC OBFUSCATION SYSTEM

We describe here our basic obfuscation system, which follows the approach presented in [2], integrated with [7] for adversaries holding map information, and with [3] for measurement imprecision.

From now on, the notation:

$$A = \text{circle}(\mathbf{X}, r)$$

will mean that A is a circle with center $\mathbf{X}$ and radius r. The *real position* of the user is a point $\mathbf{X} \in \mathbb{R}^2$. The *measured position* is a point $\mathbf{X}_m$. The *error radius* (r_m) quantifies the precision of the positioning technology. The circle $A_m = \text{circle}(\mathbf{X}_m, r_m)$ contains the real position, and it is called the *measurement area*. We suppose that the system knows the error radius. If the positioning technology does not give this information, the system can suppose an error radius, basing on the average precision of that technology.

By means of the obfuscation process, the user's position is hidden inside an *obfuscation area*, with a larger size than the measurement area. The position of a user could be contemporaneously accessed by many LBSs, and the user may require different levels of privacy for them. For example, more obfuscation could be suitable for less trusted services or for services requiring less precision.

A solution is to *independently* generate many obfuscation areas with different size, and store them on a *location server*. The location server will in turn release to the *service provider* the obfuscation area corresponding to its access rights. With n levels of privacy, we generate n obfuscation areas:

$$A_o^{(k)} = \text{circle}(\mathbf{X}_o^{(k)}, r_o^{(k)}) \quad \text{with } 0 \le k \le n-1 \tag{1}$$

where k is the *precision index*. The higher the precision index is, the more precise the obfuscation area, and the lower the privacy level. More trusted service providers will be allowed to access to obfuscation areas with higher-precision indexes. We consider the measurement area itself as the obfuscation area with precision index n:

$$A_o^{(n)} = A_m \tag{2}$$

So we actually have $n+1$ precision indexes, from 0 to n. The point $\mathbf{X}_o^{(k)}$ is the *kth obfuscated position*, the radius $r_o^{(k)}$ is the *kth obfuscation radius* and the area $A_o^{(k)}$ is the *kth obfuscation area*. The obfuscation radii follow a decreasing progression:

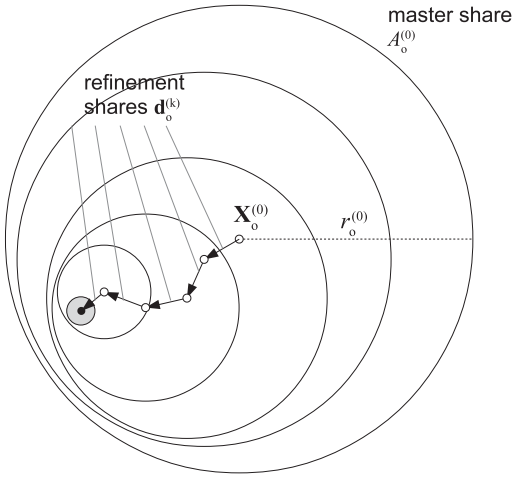

FIGURE 1. Obfuscation scheme.

$$r_o^{(k)} = \begin{cases} r_o^{(0)}/n \cdot (n-k) & \text{if } 1 \le k \le n-1 \\ r_m & \text{if } k = n \end{cases} \tag{3}$$

This solution is simple and scalable. However, if the obfuscation areas are stored in a single location server, the user is forced to trust such a server. We solve this problem by splitting the whole set of obfuscation areas in pieces (*shares*), and then storing them in several location servers, each of which is not required to be trustworthy. The obfuscated positions are randomly generated by means of n concatenated random vectors called *obfuscation vectors* ($\mathbf{d}_o^{(k)}$). The obfuscation vectors form a chain that connects all the obfuscated positions, from the zeroth one to the measured position (Fig. 1). The largest obfuscation area constitutes the *master share*, and the n obfuscation vectors the *refinement shares*. The kth obfuscation area can be reconstructed by combining the master share with the first k refinement shares. The share combination is done by reducing the obfuscation radius (following Equation (3)) and composing the obfuscation vectors:

$$\mathbf{X}_o^{(k)} = \mathbf{X}_o^{(0)} + \sum_{i=1}^{k} \mathbf{d}_o^{(k)} \tag{4}$$

The user generates the master share and the refinement shares, and distributes these $n+1$ pieces of information to $n+1$ location servers (Fig. 2). To grant a service provider the access to the kth obfuscation area, the user simply grants him access to the first $k+1$ location servers. The service provider retrieves the correspondent shares, and compose them to obtain back the obfuscation area.

This obfuscation system enjoys the property that neither the service providers nor the location servers have to be trusted by the user, as no one of these entities has the complete set

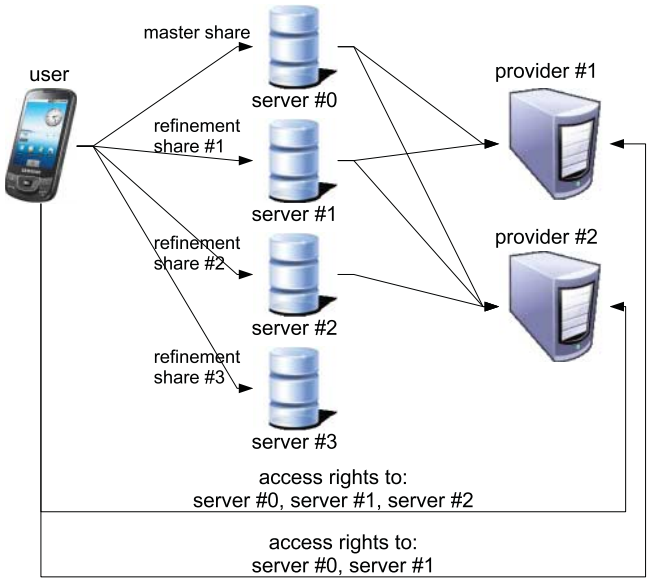

FIGURE 2. Architecture for multiple levels of privacy.

of information. If the master share and k refinement shares get compromised, the adversary will know the position at the precision index k at most. In other words, the user's privacy degrades gracefully with the number of compromised shares.

Note that a privacy improvement has always an impact on the quality of the provided service. In other words, by obfuscating a position we inevitably reduce its *utility*, which depends in general on the obfuscation radius. The larger it is, the less the utility is expected to be. Though evaluation of the utility loss is important, however, it falls beyond the scope of this paper, which is about how to make obfuscators more resilient to statistical attacks while maintaining their utility, i.e. the obfuscation radius. For an insightful view of utility aspects in the presence of a generic obfuscation technique, readers may refer to [20].

3.1. Architectural considerations

Our system achieves the stated goals by utilizing multiple location servers of different providers. This option is made possible by the emerging technology trend of building services upon federated systems [21, 22], which not only prevents provider lock-in, but also offers more resources to be used. The increasing availability of large distributed infrastructures at a reasonable price (including cloud-based ones) provides scalable and efficient management of large amounts of location data and supports efficient query processing over this data. The appropriate infrastructures are already offered by major providers, for example, Amazon [23], Google [24], Microsoft, IBM and some smaller companies like ElasticHosts [25], Rackspace [26], XCalibre Communications [27].

In terms of efficiency, the traffic generated by a single location update increases, due to the fact that the user has to communicate with $n + 1$ servers instead of one. However, the total traffic is quite negligible compared with the capacity of modern cellular data connections. A master share can be represented by a regular GPS position, which in Android systems is two doubles for the polar coordinates (8 bytes each) and a float for the radius (4 bytes). A refinement share can be represented by two floats for the X and the Y components (4 bytes each). With these tiny payloads, the traffic amount generated by a location update is dominated by the establishment of the secure sessions [28], in which the servers send their certificates to the client. In the case of SSL protocol, the total traffic for a session establishment is ∼1810 bytes (183 bytes uplink and 1627 bytes downlink [28]). Supposing $n = 5$ levels of privacy (thus six location servers), the total traffic generated by a location update is ∼10.6 kb. A user performing 20 location updates per day will spend ∼6.21 Mb per month. This rough computation does not claim to be fully realistic, since in the real-life operation there will be further traffic costs (packet losses, etc.) and some traffic savings (SSL session reuses, etc.). However, it gives the order of magnitude of the generated traffic, which is fully sustainable by modern cellular connections.

3.2. Share generation methods

In [2], we used two possible methods to generate the shares: *a posteriori share generation method*, and *a priori share generation method*. The *a posteriori* one first generates the refinement shares, and then the master share. The *a priori* one does vice versa.

Both methods use the concept of *r-bounded uniform vector* as a fundamental building block.

DEFINITION 3.1. *An* r-bounded uniform vector (Fig. 3) *is a random vector* $\mathbf{d}$ *such that* $\|\mathbf{d}\| \leq r$, *and its spacial probability distribution is uniform inside* $\text{circle}(\mathbf{O}, r)$, *where* $\mathbf{O}$ *indicates the axis origin.*

In the *a posteriori* method, the obfuscation vectors are generated as n independent random vectors. Then, the zeroth obfuscated position is computed by subtracting them to the measured position.

The random vectors are bounded uniform vectors, whose bounds are the following:

$$\|\mathbf{d}_o^{(k)}\| \leq \begin{cases} r_o^{(0)}/n & \text{if } 1 \leq k \leq n-1 \\ r_o^{(0)}/n - r_m & \text{if } k = n \end{cases} \quad (5)$$

Note that, on the contrary of the original algorithm from [2], the last obfuscation vector is bounded by $r_o^{(0)}/n - r_m$ (instead of $r_o^{(0)}/n$). Otherwise, depending on the measurement error,

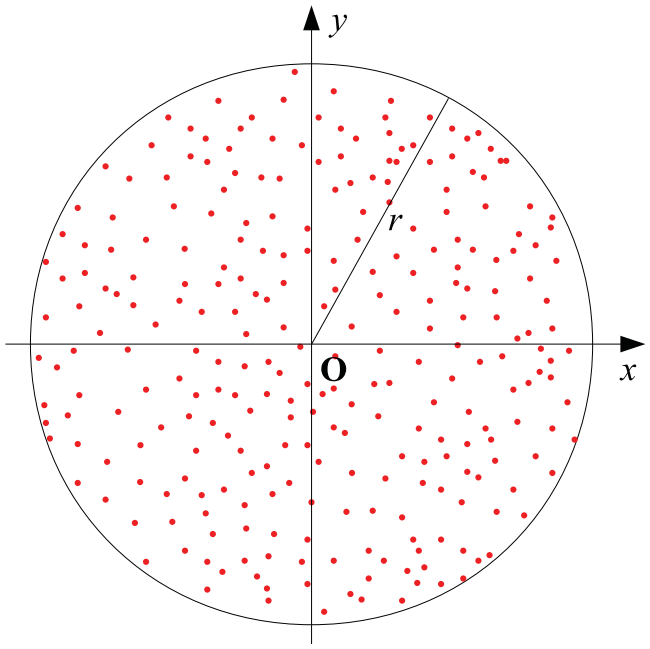

FIGURE 3. Spacial probability distribution of an r-bounded uniform vector.

Algorithm 1 *A posteriori* share generation method.

1: **procedure** APOSTERIORI($A_m, r_o^{(0)}, n$)
2: **for** $i = 1 \rightarrow (n-1)$ **do**
3: $\mathbf{d}_o^{(i)} \leftarrow (r_o^{(0)}/n)$**-bounded uniform vector**
4: **end for**
5: $\mathbf{d}_o^{(n)} \leftarrow (r_o^{(0)}/n - r_m)$**-bounded uniform vector**
6: $\mathbf{X}_o^{(0)} \leftarrow \mathbf{X}_m - \sum_{i=1}^{n} \mathbf{d}_o^{(i)}$
7: **return** $\left\{A_o^{(0)}, \mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\}$
8: **end procedure**

the real position could lie outside the obfuscation areas. Algorithm 1 shows the *a posteriori* share generation method.

The advantage of this algorithm is that the obfuscation vectors are generated independently, so that an adversary who knows some of them takes no advantage in predicting the other ones. The main drawback is that the probability density of the user position inside the zeroth obfuscation area is very biased, as shown in [2].

In the *a priori* method, we first generate the zeroth obfuscation area by means of an $(r_o^{(0)} - r_m)$-bounded uniform vector $\mathbf{d}_o^*$, called *master obfuscation vector*. Note that, on the contrary of the original algorithm from [2], the master obfuscation vector is bounded by $r_o^{(0)} - r_m$ (instead of $r_o^{(0)}$). Otherwise, depending on the measurement error, the real position could lie outside the obfuscation area. Then, we generate the refinement shares by means of a *random decomposition* of the master obfuscation vector. More formally:

DEFINITION 3.2. *Given* $\mathbf{d}_o^*$ *such that* $\|\mathbf{d}_o^*\| \leq r_o^{(n)} - r_m$, *we call* random decomposition *of* $\mathbf{d}_o^*$ *a set of random subvectors*:

$$\{\mathbf{d}_o^{(1)}, \mathbf{d}_o^{(2)}, \ldots, \mathbf{d}_o^{(n)}\}$$

such that

$$\sum_{i=1}^{n} \mathbf{d}_o^{(i)} = \mathbf{d}_o^* \tag{6}$$

and

$$\|\mathbf{d}_o^{(k)}\| \leq \begin{cases} r_o^{(0)}/n & \text{if } 1 \leq k \leq n-1 \\ r_o^{(0)}/n - r_m & \text{if } k = n \end{cases} \tag{7}$$

Algorithm 2 shows an efficient way to implement the random decomposition with bounded uniform subvectors. It generates the first $n-1$ vectors in such a way, at each step, the remaining distance (l) is coverable by the remaining vectors (Algorithm 2 Line 6). The last vector is generated as a difference, to reach exactly the master obfuscation vector (Line 7). Algorithm 3 shows the *a priori* share generation method.

Algorithm 2 Random decomposition.

1: **procedure** DECOMPOSE($\mathbf{d}_o^*, r_m, r_o^{(0)}, n$)
2: $\mathbf{d}_{sum} = 0$
3: **for** $i = 1 \rightarrow (n-1)$ **do**
4: $l \leftarrow (n-i)r_o^{(0)}/n - r_m$
5: $\mathbf{d}_o^{(i)} \leftarrow (r_o^{(0)}/n)$**-bounded uniform vector,**
6: **such that:** dist $\left(\mathbf{d}_{sum} + \mathbf{d}_o^{(i)}, \mathbf{d}_o^*\right) \leq l$
7: $\mathbf{d}_{sum} \leftarrow \mathbf{d}_{sum} + \mathbf{d}_o^{(i)}$
8: **end for**
9: $\mathbf{d}_o^{(n)} \leftarrow \mathbf{d}_o^* - \mathbf{d}_{sum}$
10: **return** $\left\{\mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\}$
11: **end procedure**

Algorithm 3 *A priori* share generation method.

1: **procedure** APRIORI($A_m, r_o^{(0)}, n, M$)
2: $\mathbf{d}_o^* \leftarrow (r_o^{(0)} - r_m)$**-bounded uniform vector**
3: $\mathbf{X}_o^{(0)} \leftarrow \mathbf{X}_m - \mathbf{d}_o^*$
4: $\left\{\mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\} \leftarrow$ decompose($\mathbf{d}_o^*, r_m, r_o^{(0)}, n$)
5: **return** $\left\{A_o^{(0)}, \mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\}$
6: **end procedure**

The main advantage of this algorithm is that the zeroth obfuscated position is generated directly, and not as a sum

of previously generated random vectors. Thus, it is possible to control it in such a way that the real user position gets uniformly distributed. The disadvantage is that the obfuscation vectors generated by decomposition are not probabilistically independent of each other. Thus, an adversary knowing one or more of them is helped in predicting the others.

In summary, the *a priori* method is less resistant in case of more powerful adversaries, which already know some refinement shares. The *a posteriori* method is less resistant in case of less powerful adversaries, which know zero or few refinement shares.

3.3. Enlarge-and-perturb method for map awareness

Until now, we took into consideration a user who moves completely free in space. In the real life, people's movements are constrained by the presence of walls, buildings and other obstacles. An adversary who owns information about the map where the users are moving is more powerful. She can cut away from the obfuscation area the zones where the user cannot be, thus finding a *map-reduced obfuscation area*, with a smaller size. In order to guarantee a nominal level of privacy in the presence of a map-aware adversary, the obfuscation system must be map-aware too.

For our present purposes, a *map* (M) is a subset of the $\mathbb{R}^2$ space. A point is inside the map if and only if it represents a possible position for the user. For the sake of simplicity, let us consider by now a single level of privacy ($n = 1$). We will generalize to multiple levels afterwards. Let us suppose that $r_o^{(0)}$ is the 'nominal' obfuscation radius desired by the user. The obfuscation system generates an obfuscation area ($A_o^{(0)}$) whose size ($(r_o^{(0)})^2 \cdot \pi$) is the *nominal obfuscation precision*. The adversary computes a *map-reduced obfuscation area*:

$$A_M^{(0)} = A_o^{(0)} \cap M \tag{8}$$

which contains the real position of the user. In doing so, the adversary improves her precision with respect to the nominal obfuscation precision:

$$\text{size}(A_M^{(0)}) \leq (r_o^{(0)})^2 \cdot \pi \tag{9}$$

We have thus to compensate somehow this 'precision gain'.

In [7], we used a simple solution for this that we call here *enlarge-and-perturb*. Enlarge-and-perturb technique first enlarges the obfuscation radius, in such a way that the size of the map-reduced obfuscation area is equal to the nominal one. Algorithm 4 shows an efficient way to do that. We employ a logarithmic search to minimize the number of area intersections (Algorithm 4, Line 10). The logarithmic search stops when the map-reduced obfuscation area reaches the nominal

Algorithm 4 Radius enlargement algorithm.

```
1:  procedure ENLARGE(X_o, r_o, M)
2:      r_lo ← r_o
3:      r_hi ← r_o
4:      repeat                          ▷ search for a higher bound to r'_o:
5:          r_hi ← √2 · r_hi
6:          A_M ← circle(X_o, r_hi) ∩ M
7:      until size(A_M) ≥ r_o^2 · π
8:      loop                            ▷ logarithmic search for r'_o:
9:          r_md ← √((r_lo^2 + r_hi^2)/2)
10:         A_M ← circle(X_o, r_md) ∩ M
11:         if |size(A_M) − r_o^2 · π| ≤ δ_A then
12:             return r_md
13:         else if size(A_M) < r_o^2 · π then
14:             r_lo ← r_md
15:         else
16:             r_hi ← r_md
17:         end if
18:     end loop
19: end procedure
```

FIGURE 4. Enlarge-and-perturb technique.

size, with a tolerance (δ_A), that we fixed to be 1% of the nominal size:

$$\delta_A = 0.01 \cdot r_o^2 \cdot \pi \tag{10}$$

We indicate with $r_o'^{(0)}$ and $A_o'^{(0)}$, respectively, the *enlarged obfuscation radius* and the *enlarged obfuscation area*.

Enlarging the radius is obviously not enough, as the adversary could simply narrow it again. The second phase is in fact to *perturb* the obfuscated position with an ($r_o'^{(0)} - r_o^{(0)}$)-bounded uniform vector. Figure 4 shows an example of enlarge-and-perturb operation. $\mathbf{d}_p$ is the perturbation vector, while $\mathbf{X}'_o$ is the new obfuscated position after the perturbation.

Note that, after having changed the obfuscated position, the obfuscation area has changed as well. Therefore, we have to check again if the size of the map-reduced obfuscation area has become smaller. If it has, we perform an additional enlargement and an additional perturbation, and so on. In case of $n > 1$ levels of privacy, we repeat the whole procedure for each level. Algorithm 5 shows the complete enlarge-and-perturb technique.

Algorithm 5 Enlarge-and-perturb technique.

procedure ENLARGE-AND-PERTURB($A_o^{(0)}, \mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}, M$)
  $S \leftarrow \{1, \ldots, n\}$
  for all i**:** $r_o'^{(i)} \leftarrow r_o^{(i)}$ **and** $\mathbf{X}_o'^{(i)} \leftarrow \mathbf{X}_o^{(i)}$
  repeat
    for all $i \in S$ **do** ▷ Enlargement:
      $r_o'^{(i)} \leftarrow \text{enlarge}(\mathbf{X}_o'^{(i)}, r_o'^{(i)}, M)$
    end for
    for $i = 1 \rightarrow n$ **do** ▷ Perturbation:
      $\mathbf{d}_p \leftarrow (r_o'^{(i)} - r_o^{(i)})$**-bounded uniform vector**
      $\mathbf{d}_o'^{(i)} \leftarrow \mathbf{d}_o^{(i)} + \mathbf{d}_p$
    end for
    for all i**: compute** $\mathbf{X}_o'^{(i)}$ **and** $A_o'^{(i)}$ **from** $\mathbf{d}_o'^{(i)}$
    $S \leftarrow \left\{ i : \text{size}\left(A_o'^{(i)} \cap M\right) < \pi \cdot \left(r_o^{(i)}\right)^2 \right\}$
  until $S = \emptyset$
  return $A_o'^{(0)}, \mathbf{d}_o'^{(1)}, \ldots, \mathbf{d}_o'^{(n)}$
end procedure

4. ADVERSARY MODEL

We consider an adversary who wants to *deobfuscate* a previously obfuscated position, and derive the original position of the user from it. An obfuscation function cannot be deterministically inverted, since it involves random noise. However, the adversary can perform a *statistical analysis*, to find out the spatial probability distribution of the real position inside the obfuscation area. If such a distribution is uniform, the user's position is unpredictable. Otherwise, if it presents pronounced concentrations (Fig. 5), the adversary can suppose the user's position is more likely to be in certain zones than in others.

In order to measure the resistance against a statistical analysis, we have to quantify the unpredictability of the user's position, i.e. the uniformity of its probability distribution. Let us suppose that the adversary identifies an area (*deobfuscation area*), which contains the user with a certain probability (*deobfuscation probability*).

The best strategy for the adversary is to choose the deobfuscation area comprising the zones with the highest probability concentrations. We fix the size of the deobfuscation area to be 10% of the size of the obfuscation area. Figure 6 shows a single-dimension analogy. The outer and the inner segments represent, respectively, the obfuscation area and the deobfuscation area. The dashed area under the probability distribution represents the deobfuscation probability. The adversary is free to move the deobfuscation area in order to maximize the deobfuscation probability. Our resistance metric is the *maximal deobfuscation probability* (P_{deobf}).

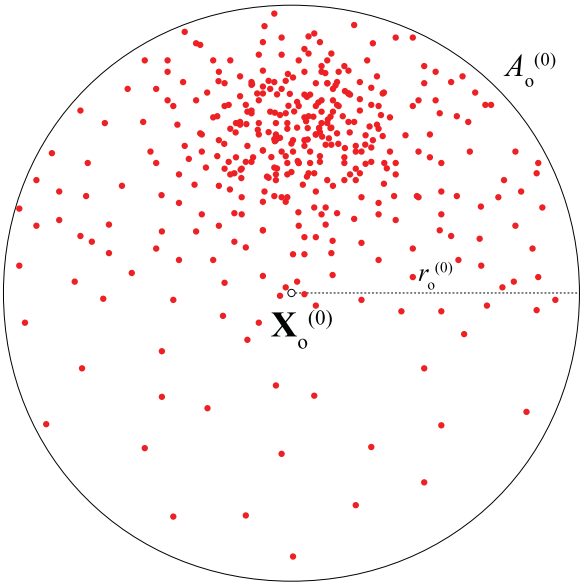

FIGURE 5. Statistical analysis example.

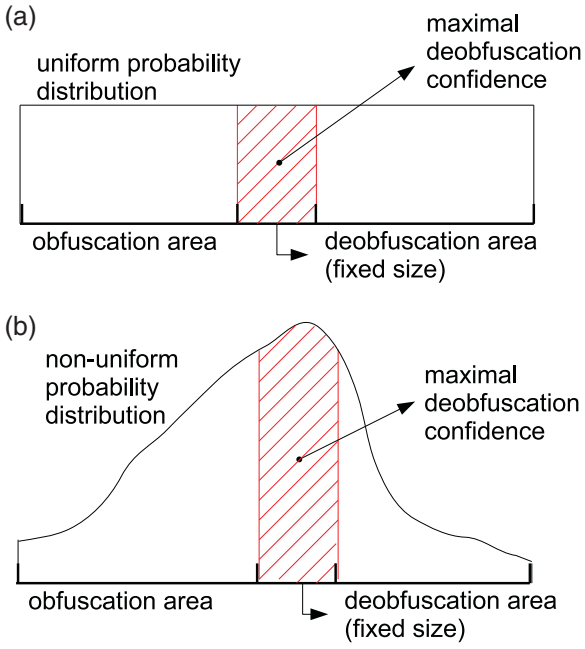

FIGURE 6. Single-dimension analogy of deobfuscation area.

DEFINITION 4.1. *Given an obfuscation area* $A_o^{(k)}$, *the* maximal deobfuscation probability *is*

$$P_{\text{deobf}} = \max_{A_o^{10\%}} Pr\left[\mathbf{X} \in A_o^{10\%}\right] \tag{11}$$

where $A_o^{10\%}$ *is a deobfuscation area such that*

$$\text{size}(A_o^{10\%}) = 10\% \cdot \text{size}(A_o^{(k)}) \tag{12}$$

Such a metric depends on the probability distribution of the user's position inside the obfuscation area. If it is uniform, like in Fig. 6a, then the maximal deobfuscation probability will be minimal, i.e. it will be exactly 10%, since we fixed the deobfuscation area to be 10% of the obfuscation area. This is the best case, when the real position is completely unpredictable inside the obfuscation area. If the probability density is not uniform, like in Fig. 6b, the adversary will have a maximal deobfuscation probability >10%. The bigger the maximal deobfuscation probability is, the more predictable the user's position.

4.1. Malicious provider and malicious server

We model two possible kinds of adversary: the *malicious provider* and the *malicious server*. The malicious provider models a service provider which illegitimately tries to gain more precision than she is permitted to have. In this case, the adversary knows the master share and a set of $k_{\text{adv}} < n$ refinement shares she has the right to access. The refinement shares get compromised in order, i.e. $\mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(k_{\text{adv}})}$, because this is the order in which the user grants the access to them. In case of two or more colluding service providers, they will be modeled as a single adversary enjoying the widest access privilege among the colluding entities. The malicious provider first combines the shares and obtains the k_{adv}th obfuscation area. Then she tries to deobfuscate such an area by means of statistical analysis.

On the contrary, the malicious server models a location server or a group of colluding location servers that want to use the shares they are storing for illegitimate purposes. It models also an external adversary who hacks one or more location servers and steals their shares. We consider the master share to be always compromised, otherwise no statistical attack is possible. In addition, we assume that $k_{\text{adv}} < n$ refinement shares are compromised, *with no particular order*. In this case, the adversary could miss one or more obfuscation vectors necessary to reconstruct the k_{adv}th obfuscation area. However, we assume that she composes anyway the obfuscation vectors she has, thus obtaining an *alternative obfuscation area* A_o^*. If we call $\mathcal{D}$ the set of the compromised obfuscation vectors, the adversary computes the alternative obfuscation area in the following way:

$$A_o^* = \text{circle}(\mathbf{X}_o^*, r_o^*) \tag{13}$$

$$\mathbf{X}_o^* = \mathbf{X}_o^{(0)} + \sum_{\mathcal{D}} \mathbf{d}_o^{(i)} \tag{14}$$

$$r_o^* = \begin{cases} r_o^{(0)}/n \cdot (n - k_{\text{adv}}) + r_m & \text{if } \mathbf{d}_o^{(1)} \in \mathcal{D} \\ r_o^{(0)}/n \cdot (n - k_{\text{adv}}) & \text{otherwise} \end{cases} \tag{15}$$

From the geometrical properties of the obfuscation vectors, the alternative obfuscation area contains the user's position. We assume the adversary performs the statistical analysis over this obfuscation area.

5. EXTREME VECTORS

As we said in Section 3, *a posteriori* and *a priori* share generation methods both have some drawbacks that limit the unpredictability of the obfuscation. In particular, the drawback of *a posteriori* share generation method is that the probability density of the real position inside the zeroth obfuscation area is very biased. Indeed, since we add n independent random vectors to generate it, the zeroth obfuscated position will follow the law of the Central Limit Theorem. As n grows, the real position of the user will tend to follow a Gaussian probability distribution inside the zeroth obfuscation area. Moreover, the larger n is, the more concentrated the probability at the center of the area will be. As $n \to \infty$, the real position's probability distribution tends to be a Dirac delta.

On the other hand, the drawback of *a priori* share generation method is that the generated obfuscation vectors are not probabilistically independent of each other. In fact, especially if the length of the master obfuscation vector is close to the limit $r_o^{(0)} - r_m$, the subvectors will be constrained to be long and to follow the same direction (Fig. 7). In other words, the

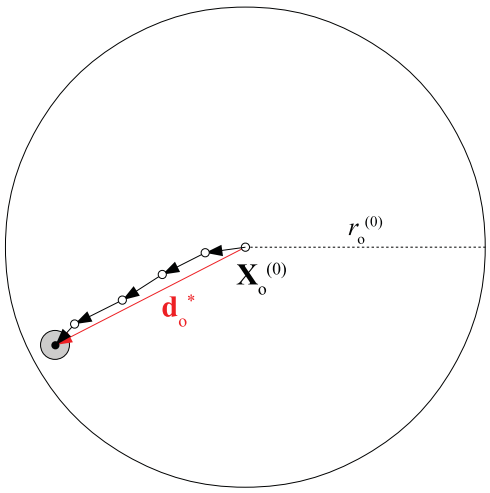

FIGURE 7. Constrained random decomposition.

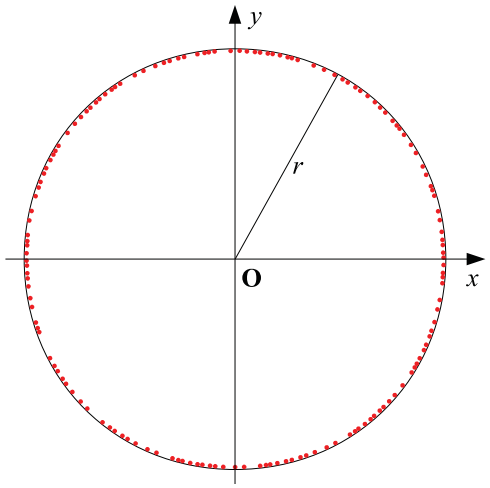

FIGURE 8. Spacial probability distribution of an r-bounded extreme vector.

obfuscation vectors are *correlated*. An adversary knowing one or more of them is helped in predicting the others.

We now introduce *extreme vectors*, an alternative to classic uniform vectors which significantly alleviates both these drawbacks, thus improving the uniformity of both obfuscation algorithms.

Definition 5.1. *An* r-bounded extreme vector (Fig. 8) *is a random vector* $\mathbf{d}$ *such that* $\|\mathbf{d}\| = r$, *and its spacial probability distribution is uniform on the border of* circle$(\mathbf{O}, r)$.

Note that extreme vectors by themselves are more predictable than uniform ones, because they are distributed on the circumference instead of inside the whole circle. However, they enjoy two good properties:

(1) *Uniform composition.* A sum of extreme vectors is less predictable than a sum of uniform vectors. This property can improve the resistance of *a posteriori* share generation method.
(2) *Uncorrelated decomposition.* A random decomposition in extreme subvectors is less correlated than a random decomposition in uniform vectors. This property can improve the resistance of *a priori* share generation method.

The reason for Property (1) is that extreme vectors help spreading the probability distribution toward the borders of the obfuscation area. This avoids the concentration at the center, thus improving the uniformity. More formally, an extreme vector has a standard deviation ($\sigma = 1/\sqrt{2}$) higher than those of a uniform vector ($\sigma = \frac{1}{2}$). Thus, a sum of n extreme vectors will have a higher standard deviation too, and will converge more slowly to a Dirac delta function. Table 1 shows the maximal deobfuscation probability of a sum of n uniform vectors and n extreme vectors.[1] It can be seen that the sum of two or more extreme vectors is more uniform.

TABLE 1. Maximal deobfuscation probability of sums of vectors.

Vector kind:	$n=1$	$n=2$	$n=3$	$n=4$
Uniform (%)	10	29.36	42.60	53.18
Extreme (%)	100	20.54	26.78	29.22
Vector kind:	$n=5$	$n=6$	$n=7$	$n=8$
Uniform (%)	62.12	69.19	75.02	79.80
Extreme (%)	37.49	43.33	48.56	53.87

The reason for Property 5 is that extreme vectors contain less information by their own. An adversary who knows one of the subvectors has no real advantage in guessing the other ones. The fact that one subvector is long is not as informative as in the case of uniform vectors. The output of a decomposition in extreme vectors is correlated as well, but to a lesser extent. We will call a random decomposition in extreme subvectors *extreme random decomposition*. Algorithm 6 shows an efficient way to implement it. Note that the last subvector is not extreme, since it is computed as a difference.

Algorithm 6 Extreme random decomposition.

procedure X-DECOMPOSE($\mathbf{d}_o^*, r_m, r_o^{(0)}, n$)
  $\mathbf{d}_{sum} = 0$
  for $i = 1 \to (n-1)$ **do**
    $l \leftarrow (n-i)r_o^{(0)}/n - r_m$
    $\mathbf{d}_o^{(i)} \leftarrow (r_o^{(0)}/n)$**-bounded extreme vector,**
    such that: dist $\left(\mathbf{d}_{sum} + \mathbf{d}_o^{(i)}, \mathbf{d}_o^*\right) \leq l$
    $\mathbf{d}_{sum} \leftarrow \mathbf{d}_{sum} + \mathbf{d}_o^{(i)}$
  end for
  $\mathbf{d}_o^{(n)} \leftarrow \mathbf{d}_o^* - \mathbf{d}_{sum}$
  return $\left\{\mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\}$
end procedure

We use the Pearson's correlation coefficient to measure the degree of correlation. Such a coefficient is defined as

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y} \tag{16}$$

where X and Y are two random variables, $\mathrm{Cov}(X, Y)$ is their covariance, and σ_X and σ_Y are their standard deviations. The closer the coefficient is to zero, the less correlated the two variables. The following matrix contains the Pearson's

[1] Each estimation stems from 100 000 vector sums.

correlation coefficients between the subvectors of a random decomposition[2]:

$$(\rho_{\mathbf{d}_o^{(i)},\mathbf{d}_o^{(j)}}) = \begin{pmatrix} 1.00 & & & & \\ 0.33 & 1.00 & & - & \\ 0.33 & 0.61 & 1.00 & & \\ 0.32 & 0.60 & 0.80 & 1.00 & \\ 0.32 & 0.60 & 0.79 & 0.91 & 1.00 \end{pmatrix}$$

The (i, j)th element contains the Pearson's correlation coefficient between the ith and the jth subvectors. It can be seen that the first subvector is relatively little correlated to the successive (first column). On the other hand, the last subvectors are very correlated to each other (second to fourth columns), because they are constrained to be long and follow the same direction. The following is the analogous matrix for an extreme random decomposition:

$$(\rho_{\mathbf{d}_o^{(i)},\mathbf{d}_o^{(j)}}) = \begin{pmatrix} 1.00 & & & & \\ 0.21 & 1.00 & & - & \\ 0.23 & 0.43 & 1.00 & & \\ 0.23 & 0.45 & 0.62 & 1.00 & \\ 0.23 & 0.44 & 0.60 & 0.71 & 1.00 \end{pmatrix}$$

It can be seen that the correlation coefficients are always lower. Thus, an extreme decomposition produces less correlated vectors.

In the following, we will show how to employ extreme vectors in *a posteriori* and in *a priori* share generation methods. We will refer to these unimproved methods as *vanilla* versions. The modified versions, based on extreme vectors, will be the *extreme* versions. Algorithm 7 shows the *extreme a posteriori share generation method*. Note that the first $n-1$ refinement shares are extreme vectors, while the last one is a uniform vector. This is in order to maintain the uniformity of the $(n-1)$th obfuscation area. Algorithm 8 shows the *extreme a priori share generation method*.

Algorithm 7 Extreme *a posteriori* share generation method.

1: **procedure** X-APOSTERIORI($A_m, r_o^{(0)}, n$)
2: **for** $i = 1 \rightarrow n-1$ **do**
3: $\mathbf{d}_o^{(i)} \leftarrow (r_o^{(0)}/n)$**-bounded extreme vector**
4: **end for**
5: $\mathbf{d}_o^{(n)} \leftarrow (r_o^{(0)}/n - r_m)$**-bounded uniform vector**
6: $\mathbf{X}_o^{(0)} \leftarrow \mathbf{X}_m - \sum_{i=1}^{n} \mathbf{d}_o^{(i)}$
7: **return** $\left\{A_o^{(0)}, \mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\}$
8: **end procedure**

[2]Each estimation stems from 100 000 decompositions, with $n = 5$, $r_m = 10$ m and $r_o^{(0)} = 1$ km.

Algorithm 8 Extreme *a priori* share generation method.

1: **procedure** X-APRIORI($A_m, r_o^{(0)}, n$)
2: $\mathbf{d}_o^* \leftarrow (r_o^{(0)} - r_m)$**-bounded uniform vector**
3: $\mathbf{X}_o^{(0)} \leftarrow \mathbf{X}_m - \mathbf{d}_o^*$
4: $\left\{\mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\} \leftarrow$ X-decompose($\mathbf{d}_o^*, r_m, r_o^{(0)}, n$)
5: **return** $\left\{A_o^{(0)}, \mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}\right\}$
6: **end procedure**

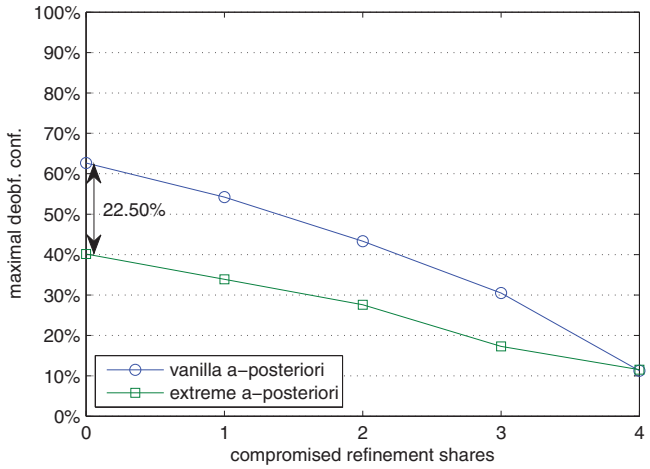

FIGURE 9. Resistance of vanilla and extreme *a posteriori* share generation methods against malicious provider.

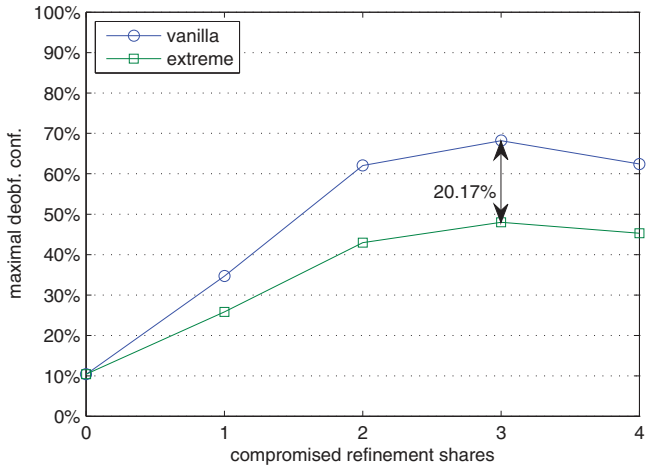

FIGURE 10. Resistance of vanilla and extreme *a priori* share generation methods against malicious provider.

5.1. Evaluation of extreme share generation methods

We evaluated the resistance of the extreme share generation methods with $n = 5$ privacy levels.

Figures 9 and 10 show the maximal deobfuscation probability of extreme *a posteriori* and *a priori* methods

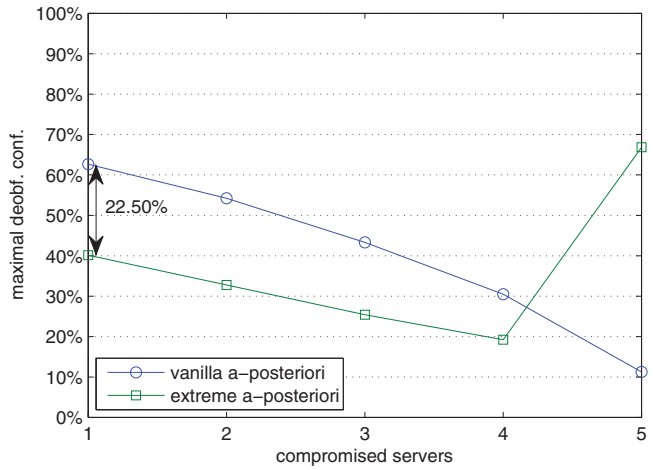

FIGURE 11. Resistance of vanilla and extreme *a posteriori* share generation methods against malicious server.

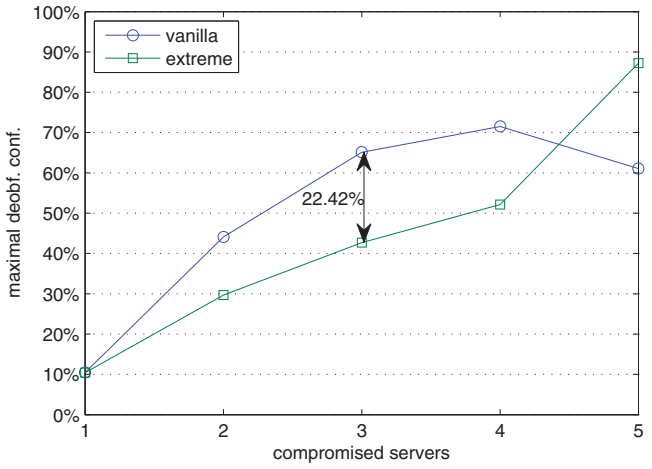

FIGURE 12. Resistance of vanilla and extreme *a priori* share generation methods against malicious server.

compared with their vanilla counterparts, against a malicious provider.[3] The master share is considered to be always compromised. On the abscissas we have the number of compromised refinement shares, i.e. the privacy level that the malicious provider has the right to access. As expected, the extreme versions are always more resistant than the vanilla versions, independently of the number of compromised refinement shares. In the *a posteriori* approach, the major improvement is in the lower privacy levels (22.50% for the zeroth privacy level). In the *a priori* approach, the resistance of the zeroth privacy level is already perfect, thus cannot be furthermore improved. However, the probabilistic correlation between the obfuscation vectors is significantly mitigated, and this has a positive effect on the resistance in case of higher privacy levels (20.17% for the third privacy level).

Figures 11 and 12 show the maximal deobfuscation probability of extreme *a posteriori* and *a priori* methods compared with their vanilla counterparts, against a malicious server. The location server holding master share is considered to be always malicious. On the abscissas we have the number of malicious servers. For instance, 'three servers' means that the server holding the master share and other two (randomly chosen) servers are malicious. As expected, the extreme versions are more resistant than the vanilla versions, except when there are $n - 1$ malicious servers. This is because the last non-compromised refinement share will be, with high probability, an extreme vector, which is poorly resilient by itself. However, this happens only in a very pessimistic case, because all the servers except one have to be malicious.

We conclude that, by using extreme vectors instead of classic uniform vectors, both share generation methods significantly improve their resistance, while maintaining their general characteristics (i.e. to be more resistant for less powerful adversaries for *a priori* and vice versa for *a posteriori*).

[3]Each estimation stems from 100 attack simulations, with $n = 5$, $r_m = 10\,\text{m}$, $r_o^{(0)} = 1\,\text{km}$. Gaussian-distributed measurement errors are assumed.

6. ENLARGE-AND-SCALE

We will now present *enlarge-and-scale*, a technique to significantly improve the uniformity of obfuscation algorithms in case of map-aware adversaries. Enlarge-and-scale is applicable to vanilla share generation methods, as well as to their extreme versions. Like enlarge-and-perturb (cfr. Algorithm 5), enlarge-and-scale first enlarges the obfuscation radius (cfr. Algorithm 4). Then, instead of perturbing the center, it performs a *scaling* of the obfuscation vector in accordance to the performed enlargement. For the sake of simplicity, let us consider by now a single level of privacy ($n = 1$). Figure 13 shows an example of enlarge-and-scale operation. $\mathbf{d}'_o$ is the scaled obfuscation vector, while $\mathbf{X}'_o$ is the new obfuscated position after the scaling operation. Like in enlarge-and-perturb, after having moved the obfuscation area, we have to check again if the size of the map-reduced obfuscation area has become smaller. If it has, we perform an additional enlargement and an additional scaling, and so on.

This enlarge-and-scale approach is preferable to the enlarge-and-perturb approach, because it avoids repeated sums of random vectors that reduce the unpredictability of the obfuscation. By scaling the existing obfuscation vectors, we do not change their probabilistic properties, and therefore we obtain the same uniformity of the map-free case.

The enlarge-and-scale technique is easily extensible in case of $n > 1$ privacy levels, both for *a posteriori* and *a priori* methods, in the following way. After the user has generated the master share and the refinement shares, he enlarges each

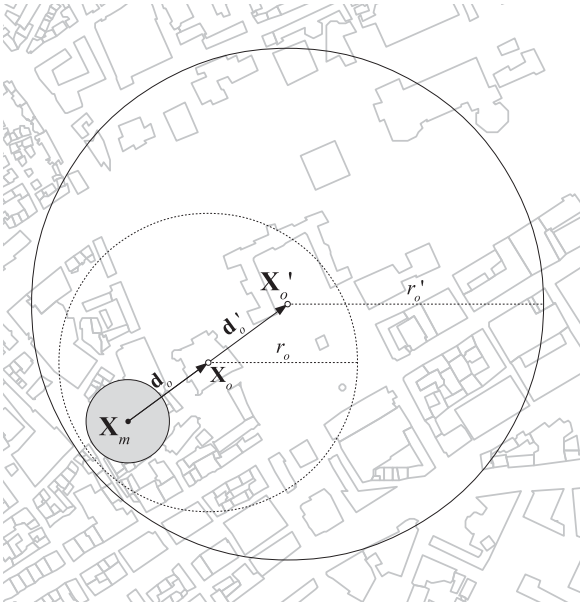

FIGURE 13. Enlarge-and-scale technique.

Algorithm 9 Enlarge-and-scale technique

1: **procedure** ENLARGE-AND-SCALE($A_o^{(0)}, \mathbf{d}_o^{(1)}, \ldots, \mathbf{d}_o^{(n)}, M$)
2: $S \leftarrow \{1, \ldots, n\}$
3: **for all** i**:** $r_o'^{(i)} \leftarrow r_o^{(i)}$ **and** $\mathbf{X}_o'^{(i)} \leftarrow \mathbf{X}_o^{(i)}$
4: **repeat**
5: **for all** $i \in S$ **do** ▷ Enlargement:
6: $r_o'^{(i)} \leftarrow \text{enlarge}(\mathbf{X}_o'^{(i)}, r_o'^{(i)}, M)$
7: **end for**
8: $\rho_{max} = \max_i \left\{ \frac{r_o'^{(i)}}{r_o^{(i)}} \right\}$
9: **for** $i = 1 \rightarrow n-1$ **do** ▷ Scaling:
10: $\mathbf{d}_o'^{(i)} \leftarrow \rho_{max} \cdot \mathbf{d}_o^{(i)}$
11: **end for**
12: $\mathbf{d}_o'^{(n)} \leftarrow \frac{\rho_{max} \cdot \mathbf{d}_o^{(n)} - r_m}{\mathbf{d}_o^{(n)} - r_m}$
13: **for all** i**: compute** $\mathbf{X}_o'^{(i)}$ **and** $A_o'^{(i)}$ **from** $\mathbf{d}_o'^{(i)}$
14: $S \leftarrow \left\{ i : \text{size}\left(A_o'^{(i)} \cap M\right) < \pi \cdot \left(r_o^{(i)}\right)^2 \right\}$
15: **until** $S = \emptyset$
16: **return** $A_o'^{(0)}, \mathbf{d}_o'^{(1)}, \ldots, \mathbf{d}_o'^{(n)}$
17: **end procedure**

obfuscation radius. Then, he chooses the largest relative radius enlargement, and he applies it to all the obfuscation areas. In the scaling phase, the user scales all the obfuscation vectors, according to the performed enlargement. Finally, he checks whether all the n map-reduced obfuscation areas have the nominal size or greater. If they have, the algorithm ends. Otherwise, the user makes an additional enlarge-and-scale step, and so on. Algorithm 9 shows the complete enlarge-and-scale technique. Note that the last obfuscation vector is scaled by a different ratio (Algorithm 9, Line 12). This is to compensate the fact that the size of the measurement area is fixed, and cannot be enlarged coherently to the other obfuscation areas.

6.1. Evaluation of enlarge-and-scale technique

We evaluated the resistance of enlarge-and-scale technique against map-aware adversaries, and we compared it with the performance of enlarge-and-perturb. We tested both algorithms on synthetic Manhattan-like maps, with square-shaped buildings, roads' width equal to 10 m, and a varying distance between parallel roads. The obfuscation areas have been created by means of extreme *a priori* share generation method. However, the enlarge-and-scale technique is independent from the share generation method, and the same results apply to the other presented methods as well. First, we want to show that enlarge-and-scale does not produce larger obfuscation areas than enlarge-and-perturb, and thus it does not decrease the quality of service. Figure 14 shows the average relative enlargement of the zeroth obfuscation area (with $n = 5$ levels of privacy) versus the ratio of free space of the map.[4] Obviously, the less the free space is, the more the obfuscation areas have to be enlarged, both in enlarge-and-scale and in enlarge-and-perturb methods. However, we can see that both methods enlarge the obfuscation areas quite equally on every map. Thus, enlarge-and-scale does not degrade the quality of service with respect to enlarge-and-perturb.

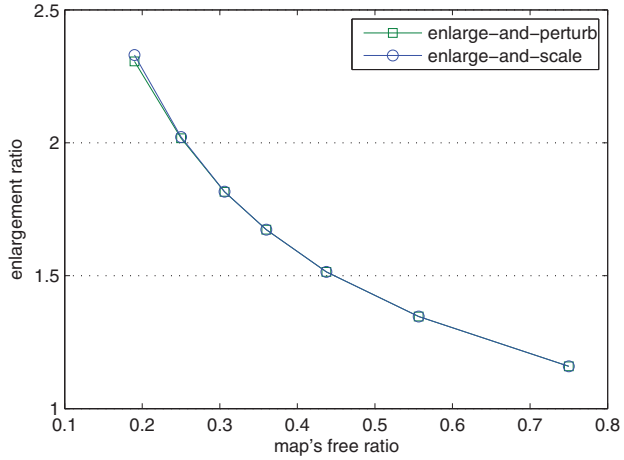

FIGURE 14. Enlargement ratio against walkable space ratio.

Regarding the deobfuscation resistance, Fig. 15 shows the maximal deobfuscation probability of a malicious provider

[4]Each estimation stems from 500 obfuscation simulations, with $n = 5$, $r_m = 10$ m, $r_o^{(0)} = 1$ km. Gaussian-distributed measurement errors are assumed.

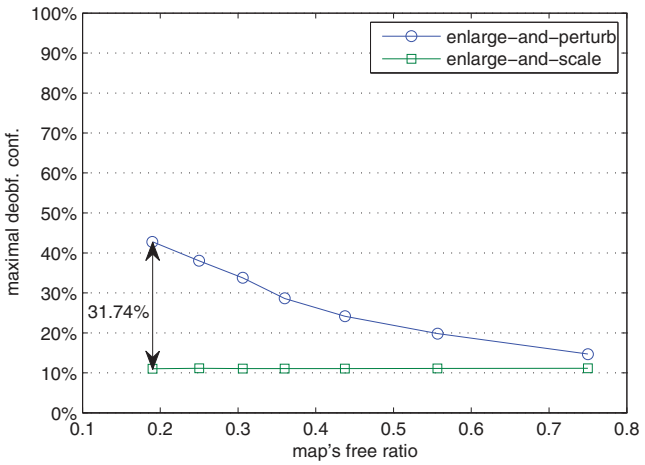

FIGURE 15. Maximal deobfuscation probability in case of map-aware adversary.

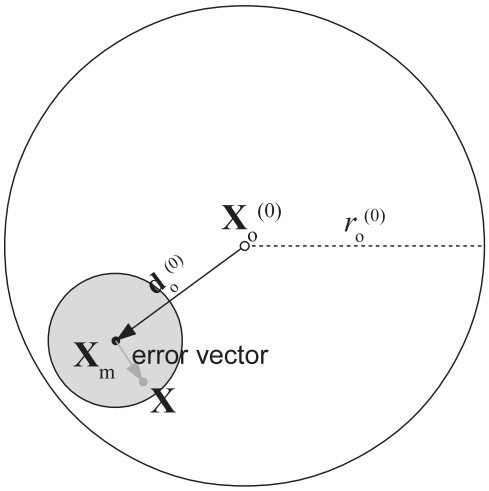

FIGURE 16. Case of non-negligible measurement error.

knowing only the master share.[5] On the abscissa we have the *map's free ratio*, i.e. the percentage of walkable space in the map. As expected, the enlarge-and-scale technique is more resistant than the enlarge-and-perturb one. In particular, it has the same uniformity expected from *a priori* share generation in free space. This is because the scaling operation does not change the probabilistic properties of the obfuscation vectors.

It can be seen that the resistance gain is particularly high for maps with a low walkable ratio (31.74% for walkable ratio equal to 0.19).

To sum up, enlarge-and-scale technique is able to significantly improve the resistance with respect to enlarge-and-perturb, because it avoids repeated sums of random vectors. This resistance gain does not cause a degradation on the quality of the service.

7. HYBRID VECTORS

Every position measurement carries with itself an error, due to technological imprecision. Different technologies have different precisions [29]. If the average position error is very small, for example below 1–5 m, as it happens in differential GPS or in UWB positioning, we can approximate it to zero. Otherwise, if the measurement noise is comparable with the obfuscation one, as it happens in smartphone's cheap GPS receivers or in cellular positioning, we cannot neglect it. In this case, the obfuscation system has to take into account the measurement imprecision in order to obfuscate in a proper way.

For the sake of simplicity, let us consider by now a single level of privacy ($n = 1$). We call *error vector* the vector $\mathbf{d}_m = \mathbf{X}_m - \mathbf{X}$ (Fig. 16), i.e. the vector between the measured and the real user's position. The error vector is a random vector over which the obfuscation system has no control. We assume that the error radius is tailored to be always longer than or equal to the error vector:

$$\|\mathbf{d}_m\| \leq r_m \tag{17}$$

In this way, the real position always lies inside the measurement area. If a technology exhibits a theoretically non-bounded error (e.g. a Gaussian one), the obfuscation system can approximate it by truncation at $r_m = 3\sigma$. In this way, only a negligible amount of measurement samples will fall outside the measurement area.

We can think the error vector as an additional 'obfuscation vector'. The system has no control over this vector, but the adversary has to deobfuscate it anyway if she wants to find the real position, which represents the true personal piece of information. As a result of the presence of an error vector, even if the obfuscation vector is uniform, the distribution of the real position will not be uniform [3]. Since the obfuscation and the error vectors constitute a sum of random vectors (cfr. Fig. 16), the extreme vectors turn out to be useful to improve the overall resistance. However, using a simple extreme vector is not convenient this time, as it produces an area with zero probability distribution at the center (dashed area in Fig. 17a). On the other hand, adding a classic uniform vector fills the hole at the center but, as shown in [3], it produces a lack of probability distribution close to the borders of the obfuscation area (dashed area in Fig. 17b).

Our idea is to use a mix between a uniform vector and an extreme one that we call *hybrid vector*. An hybrid vector depends on a real parameter $\alpha \in [0, 1]$ that we call *extremeness*.

Definition 7.1. *An* r-bounded hybrid vector *with extremeness* $\alpha \in [0, 1]$*, is an* r*-bounded uniform vector with probability* α*, or an* r*-bounded extreme vector with probability* $1 - \alpha$.

[5] Each estimation stems from 10 000 attack simulations, with $n = 5$, $r_m = 10$ m, $r_o^{(0)} = 1$ km. Gaussian-distributed measurement errors are assumed.

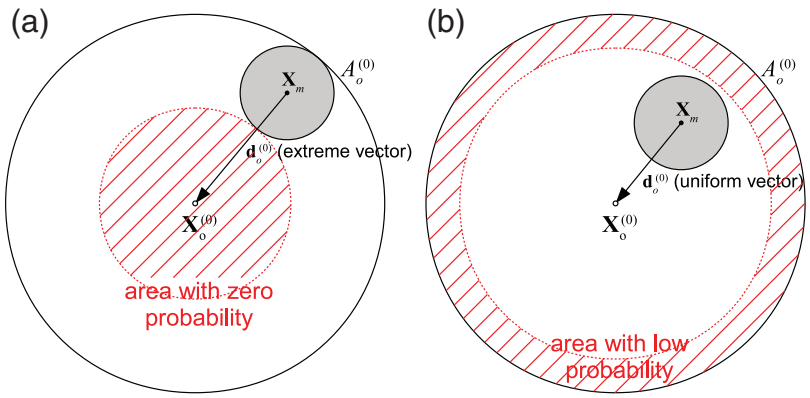

FIGURE 17. Using extreme vectors or uniform vectors for obfuscating imprecise position measurements.

Note that with an extremeness equal to 0 we obtain a uniform vector, and with an extremeness equal to 1 we obtain an extreme vector. The *optimal extremeness* (α_{opt}) is the one which maximizes the uniformity of the probability distribution, and it is somewhere in the range $[0, 1]$. It depends on the probability distribution of the error vector, and on the *radius ratio* (ρ), defined as

$$\rho = r_o^{(0)} / r_m \tag{18}$$

A radius ratio close to one indicates that the measurement imprecision is of the same magnitude order of the obfuscation noise. On the contrary, a radius ratio tending to infinite indicates that the imprecision is negligible compared with the obfuscation noise.

Computing the optimal extremeness is burdensome for a resource-constrained device, since it requires to simulate a deobfuscation attack for each value of extremeness, and then choosing the one giving the best uniformity. We propose a *heuristic extremeness* (α_{heur}) that approximates the optimal one, given the radius ratio:

$$\alpha_{\text{heur}} = \begin{cases} k_1(2\rho - k_1)/\rho^2 & \rho \in (1, \rho_1] \\ k_2(2\rho - k_2)/\rho^2 & \rho \in (\rho_1, \infty) \end{cases}$$

k_1, k_2 and ρ_1 are parameters of the heuristic. Such a heuristic function is based on geometrical considerations (Fig. 18). We divide the obfuscation area in two concentric regions: an external one (A), and an internal one (B). The external region has a width proportional to the error radius, with a proportionality constant k. Then, we make the (approximating) assumption that the real position will be in the external region if and only if the obfuscation vector is extreme (Fig. 18, upper vector), and in the internal region if and only if it is uniform (Fig. 18, lower vector). This implies that the external region contains the real position with a probability equal to the extremeness of the obfuscation vector:

$$Pr\,[\mathbf{X} \in A] = \alpha \tag{19}$$

$$Pr\,[\mathbf{X} \in B] = 1 - \alpha \tag{20}$$

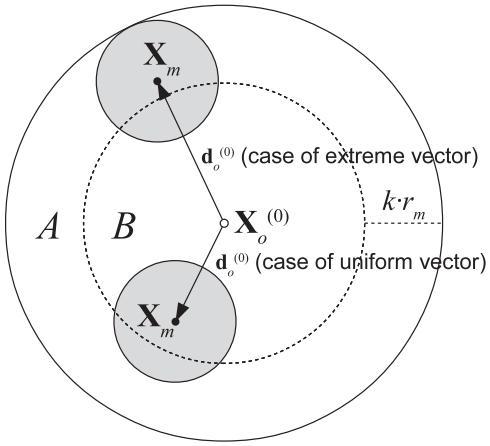

FIGURE 18. Rationale behind the heuristic.

Finally, we fix the extremeness in such a way that both regions contain a probability proportional to their size. By imposing this, we improve the overall uniformity by balancing the probability between the external and the internal regions. We used two values for k (k_1 or k_2) depending on the range in which the radius ratio lies: $(1, \rho_1]$ or (ρ_1, ∞). We noted that this makes the heuristic closer to the optimal. As the radius ratio grows, the heuristic extremeness converges to zero. This is an expected behavior because, as the measurement imprecision becomes negligible, a uniform obfuscation vector is more suitable. We consider two kinds of measurement error: (a) the *Gaussian error*, typical of GPS; and (b) the *uniform error*, typical of cellular/Wi-Fi positioning. We computed the best parameters of the heuristic for both error models, i.e. the parameters which maximize the uniformity of the probability distribution, averaged on the radius ratio. Table 2 shows the best parameters computed for the Gaussian and for the uniform error models. These values minimize the maximal deobfuscation probability, averaged on the radius ratio.

TABLE 2. Best parameters for the heuristic.

Error shape:	ρ_1	k_1	k_2
Gaussian	2.4	1.22	0.35
Uniform	3.9	1.89	0.38

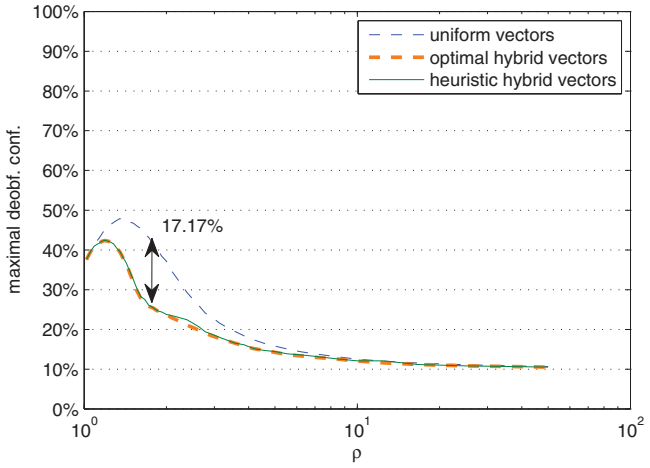

FIGURE 19. Resistance of uniform and hybrid obfuscation with Gaussian error model.

In order to employ hybrid vectors with $n > 1$ levels of privacy, it is sufficient to use them whenever an obfuscation vector is directly added to the error vector. In case of *a posteriori* share generation method, the first obfuscation vector will be generated as hybrid. In case of *a priori* share generation method, the master obfuscation vector will be generated as hybrid.

7.1. Evaluation of hybrid vectors

Figures 19 and 20 show the maximal deobfuscation probability under, respectively, Gaussian and uniform error models, obfuscated by uniform vectors, by optimal hybrid vectors, and by heuristic hybrid vectors. It can be noted that hybrid vectors always overwhelm uniform ones in terms of obfuscation uniformity. They can reduce the maximal deobfuscation probability of 17.17% under Gaussian error model, and of 13.56% under uniform error model. Also, the performance of the heuristic closely follows the optimum. In the worst case, our heuristic increases the maximal deobfuscation probability of only 1.34% under Gaussian error model, and of 0.66% under uniform error model.

To sum up, hybrid vectors are capable of significantly improving the resistance against statistical analysis in case of non-negligible measurement error. The heuristic we presented permits to compute the value of the extremeness in an efficient way, without losing resistance with respect to the optimum.

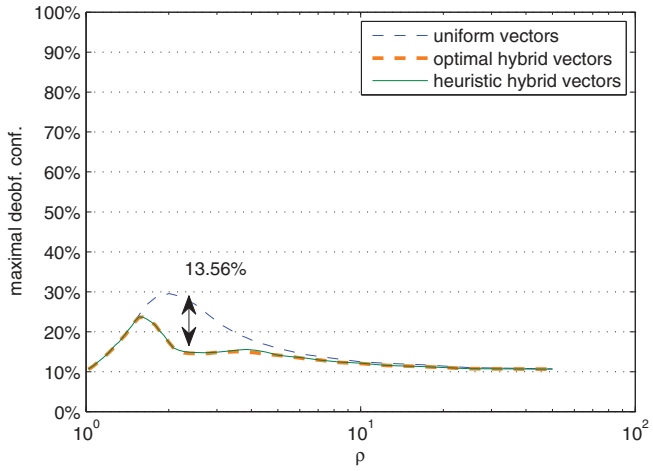

FIGURE 20. Resistance of uniform and hybrid obfuscation with uniform error model.

8. CONCLUSION

In this paper, we presented a location obfuscation system capable of dealing with measurement imprecision, multiple levels of privacy, untrusted servers and adversarial knowledge of the map. We studied its resistance against deobfuscation attacks, and we improved it by means of three techniques, namely extreme vectors, enlarge-and-scale, and hybrid vectors. Our tests showed that extreme vectors can decrease the adversarial success probability by 22.50%, enlarge-and-scale technique by 31.74% and hybrid vectors by 17.17%.

FUNDING

This work has been supported by EU FP7 Integrated Project PLANET (Grant agreement no. FP7-257649), by Italian Research Project TENACE (pr. no. 20103P34XC), and by Project PITAGORA, cofinanced by the Regional Government of Tuscany (POR CReO – Bando Unico R&S 2012) and the European Regional Development Fund (ERDF).

REFERENCES

[1] Kido, H., Yanagisawa, Y. and Satoh, T. (2005) An Anonymous Communication Technique Using Dummies for Location-Based Services. *Proc. ICPS'05*, Santorini, Greece, July 11–14, pp. 88–97. IEEE, Piscataway, NJ.

[2] Dürr, F., Skvortsov, P. and Rothermel, K. (2011) Position Sharing for Location Privacy in Non-Trusted Systems. *Proc. PerCom'11*, Seattle, WA, March 21–25, pp. 189–196. IEEE, Piscataway, NJ.

[3] Dini, G. and Perazzo, P. (2012) Uniform Obfuscation for Location Privacy. *Proc. DBSec'12*, Paris, France, July 11–13, pp. 90–105. Springer, Berlin, Heidelberg.

[4] Ardagna, C., Cremonini, M. and Damiani, E. (2007) Location Privacy Protection Through Obfuscation-Based Techniques.

Proc. DBSec'07, Redondo Beach, CA, July 8–11, pp. 47–60. Springer, Berlin, Heidelberg.

[5] Li, M., Salinas, S., Thapa, A. and Li, P. (2013) *n*-CD: A Geometric Approach to Preserving Location Privacy in Location-Based Services. *Proc. INFOCOM'13*, Turin, Italy, April 14–19, pp. 3012–3020. IEEE, Piscataway, NJ.

[6] Tennant, D. (2014) *GPS Dating app yields position-shifting technology for social media*. http://www.itbusinessedge.com/blogs/from-under-the-rug/gps-dating-app-yields-position-shifting-technology-for-social-media.html (accessed October 1, 2014).

[7] Skvortsov, P., Dürr, F. and Rothermel, K. (2012) Map-Aware Position Sharing for Location Privacy in Non-Trusted Systems. *Proc. Pervasive'12*, Newcastle, UK, June 18–22, pp. 388–405. Springer, Berlin, Heidelberg.

[8] *Foursquare*. www.foursquare.com (accessed July 20, 2013).

[9] *Loopt*. www.loopt.com (accessed July 20, 2013).

[10] *Google Latitude*. www.google.com/latitude (accessed July 20, 2013).

[11] Samarati, P. and Sweeney, L. (1998) Protecting Privacy When Disclosing Information: *k*-Anonymity and its Enforcement Through Generalization and Suppression. Technical Report. Computer Science Laboratory SRI International.

[12] Gruteser, M. and Grunwald, D. (2003) Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. *Proc. MobiSys'03*, San Francisco, CA, May 5–8, pp. 31–42. ACM, New York, NY.

[13] Chow, C.-Y., Mokbel, M.F. and Liu, X. (2006) A Peer-To-Peer Spatial Cloaking Algorithm for Anonymous Location-Based Service. *Proc. GIS'06*, Arlington, VA, November 10–11, pp. 171–178. ACM, New York, NY.

[14] Mascetti, S., Freni, D., Bettini, C., Wang, X.S. and Jajodia, S. (2011) Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *VLDB J.*, **20**, 541–566.

[15] Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C. and Tan, K.-L. (2008) Private Queries in Location-Based Services: Anonymizers Are Not Necessary. *Proc. SIGMOD'08*, Vancouver, Canada, June 9–12, pp. 121–132. ACM, New York, NY, USA.

[16] Li, M., Zhu, H., Gao, Z., Chen, S., Ren, K., Yu, L. and Hu, S. (2014) All Your Location are Belong to Us: Breaking Mobile Social Networks for Automated User Location Tracking. *Proc. MobiHoc'14*, Philadelphia, PA, August 11–14, pp. 43–52. ACM, New York, NY.

[17] WeChat. www.wechat.com (accessed September 15, 2014).

[18] MoMo. www.immomo.com (accessed September 15, 2014).

[19] Shokri, R., Theodorakopoulos, G., Troncoso, C., Hubaux, J.-P. and Le Boudec, J.-Y. (2012) Protecting Location Privacy: Optimal Strategy Against Localization Attacks. *Proc. CCS'12*, Raleigh, NC, October 16–18, pp. 617–627. ACM, New York, NY.

[20] Cheng, R., Zhang, Y., Bertino, E. and Prabhakar, S. (2006) Preserving User Location Privacy in Mobile Data Management Infrastructures. *Proc. PETS'06*, Cambridge, UK, June 28–30, pp. 393–412. Springer, Berlin/Heidelberg.

[21] Chun, B.N. and Bavier, A. (2004) Decentralized Trust Management and Accountability in Federated Systems. *Proc. HICSS'04*, Big Island, HI, Track 9, Volume 9, January 5–8. IEEE Computer Society, Washington, DC.

[22] Petri, I., Beach, T., Zou, M., Diaz-Montes, J., Rana, O. and Parashar, M. (2014) Exploring Models and Mechanisms for Exchanging Resources in a Federated Cloud. *Proc. IC2E'14*, Boston, MA, March 10–14. IEEE Computer Society, Washington, DC.

[23] Amazon Web Services LLC. aws.amazon.com (accessed October 1, 2014).

[24] Google Cloud Platform. cloud.google.com/compute (accessed October 1, 2014).

[25] ElasticHosts Ltd. www.elastichosts.com (accessed October 1, 2014).

[26] The Rackspace Cloud. www.rackspaceclout.com (accessed October 1, 2014).

[27] Flexiscale. flexiscale.com (accessed October 1, 2014).

[28] Potlapally, N., Ravi, S., Raghunathan, A. and Jha, N. (2006) A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Trans. Mob. Comput.*, **5**, 128–143.

[29] Liu, H., Darabi, H., Banerjee, P. and Liu, J. (2007) Survey of wireless indoor positioning techniques and systems. *IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev.*, **37**, 1067–1080.