LOCATION PRIVACY FOR NON-STATIONARY USERS

Iulia-Maria FLOREA[1], Dan VORNICU[2], Ștefan-Dan CIOCÎRLAN[3], Răzvan RUGHINIȘ[4]

In the last years, privacy has been a debated topic since the approval of the European GDPR. People have become more aware of the information gathered by the applications they are using often. Many of them require specific accuracy that may not necessarily need. For instance, weather-based services do not need the exact user location, since the weather is almost the same in a city. This paper focuses on location obfuscation techniques for a user moving in a city. Our solution provides a fake path, based on the real locations and we test it against state-of-the-art attacks to prove its resistance.

Keywords: location obfuscation, privacy, geo-indistinguishability

1. Introduction

Day by day, we advance deeper into the universe of technology, of devices and applications that provide services. A segment of these services are the ones that are based on user location. No matter if you are on vacation and want to check the local weather, or you need a little help finding a destination in a less frequented area, there is at least one service that, by tracking your location, can send you all the necessary information. Location-Based Services (LBS) are currently an undeniable part of everyday life ([1, 2]).

Since users do not know how these applications handle, store and use the locations, it is safer to adopt a precautionary behavior. The best-case scenario is when a user's location is shared with other entities only with the user's consent, but this is not always a guarantee. Some applications might collect user locations without them knowing [3, 4].

The need to maintain location privacy is only stronger in today's environment, when context-awareness, location-awareness services are everywhere. Current systems are able to track anyone in real time, with incredible accuracy. Together with technological advancements in processing power and capacity, the population can be followed wherever they go, much easier than ever

[1] PhD student., Dept. of Computers and Automatic Control, University POLITEHNICA of Bucharest, Romania, e-mail: iulia.florea@upb.ro
[2] MSc., Dept. of Computers and Automatic Control, University POLITEHNICA of Bucharest, Romania, e-mail: vornicu_dan2000@yahoo.com
[3] PhD Student., Dept. of Computers and Automatic Control, University POLITEHNICA of Bucharest, Romania, e-mail: stefan_dan.ciocirlan@upb.ro
[4] Prof., Dept. of Computers and Automatic Control, University POLITEHNICA of Bucharest, Romania, e-mail: razvan.rughinis@upb.ro

before. But privacy is a human right, internationally recognized. Even though this right to privacy is debated by many, there are some clear negative aspects with not respecting this right. For instance, one's location can be used maliciously to send spam messages containing products or services related to the victim's location. Another risk of ignoring location privacy is given by physical abuse. An attacker might use someone's location to follow around, rob, assault this victim. A third reason why the lack of location privacy can cause harm to a person is that the location can tell things about a person. Just by knowing where someone was in the past can lead to fair assumptions about religion, political orientation, health status and other similar private information [5]. Although it is not a universal concern, there are individuals who believe that their information could be used differently than what had been agreed upon.

In this paper, we provide a solution that creates an obfuscated, but realistic path for moving users in cities, so attackers gaining access will not be able to reconstruct the real locations. The current algorithm continues the work in [6] and it is based on an idea and a dataset included in the "Optimal Geo-Indistinguishable Mechanisms for Location Privacy" [7], a paper that concentrates on finding the balance between privacy gain and quality loss. Given the city of Beijing, China, and a map of the most frequently visited areas within the city, the initial algorithm processes the real location and outputs locations as close as possible to popular areas. We also prove that out solution is resistant to state-of-the-art attacks.

2. Related work

2.1 Real location obfuscation techniques

A similar work, "Geo-indistinguishability: A Principled Approach to Location Privacy" [3], introduces the concept of following a location trace. The authors present a way of doing so with a limited degradation of the privacy factor that is inevitable to a certain degree due to the correlation between locations. They consider an input made of an array of locations, together forming a location trace. In their view, the first and most basic solution to hide the user's path is to add noise individually to each element of the array. The next step towards a stronger solution is a prediction-based approach, that takes into account all the previous information about the trace. If the new location can be easily predicted, then it is returned without the addition of supplementary noise. They call this the easy point. A hard point cannot be predicted based on the existing data while maintaining privacy. This type of point is sanitized with noise. In their prediction step, they perform calculations in order to generate a new location at an acceptable distance from the previous point. This idea represents the foundation for the algorithm presented in the next chapter. Their solution is based on differential privacy, which leads to further computations and overhead. Our approach aims to provide a similar solution, but with a lower number of computations, so it would be suitable for embedded devices.

The same general concept of location obfuscation is also debated in a different paper, “Evaluation of Location Obfuscation Techniques for Privacy in Location Based Information Systems” [4]. The researchers compare multiple obfuscation techniques by running five algorithms on the same dataset. “Rand” is an algorithm that generates a completely random location wherever within a specified radius from the real location. “N-Rand” is a modification of the regular “Rand”, generating N random locations and selecting the one that is furthest from the real location. “N-Mix” was supposed to return a false location based on the location histories of other people, very similar with the current approach explained in this paper. However, due to the lack of data, the authors chose to generate N random points and compute the average coordinates between those points. “Dispersion” selects a random point like “Rand” does, then runs “Rand” again with a smaller radius and with the origin set on this random point. “N-Dispersion” determines the random point with the “N-Rand” algorithm, then runs “Rand” on said point with a smaller radius. The metrics of reference are based on the distance between the real and the fabricated locations. In particular, the final result is determined according to the average distance between these two types of locations after all the algorithms process a simulated route on the map. Their results pointed towards “N-Rand” as the most efficient method as long as N is not higher than 4. The “N-Rand” algorithm followed the route in a zig-zag pattern, with an average distance of approximately 25 meters. According to their analysis, the “N-Dispersion” would have also generated satisfactory results, however the complexity of this algorithm makes it inefficient compared to the others.

2.2 Attacks on location obfuscation techniques

The first type of attack is the same-origin [8]. It is based on the presumption that the user uses the same original location *i* and it generates obfuscated locations *j* with probability $p_i(j)$. The attacker is supposed to know the probability mass functions of the algorithm and so, he can build an obfuscation pattern for the several origin points in the same region of interest. By comparing the results with the observation points, he can obtain the original location. This idea was tested against several state-of-the art-implementations: spatial-cloaking, geo-indistinguishability and a maximum-entropy method that they developed. The attack success is measured in two ways. The first is the probability of getting the original location after *t* observations. The other one is the distance between the true location and the chosen location after *t* observations. The success probability is determined in their experiments after 200 observations. The k-cloaking mechanism has worse results than geo-indistinguishability and maximum-entropy solutions, that provide similar numbers.

The second type of attacks, brute force, are evaluated in [11] against two state-of-the art location obfuscation mechanisms: geo-indistinguishability [7] and expected inference error [12], [13]. It is composed of two different types, optimal inference attack [13] and Bayesian inference attack. The former tries to minimize the error between the estimated and the real locations, using the observed obfuscated location. In this case, the distance between points is Euclidian. The latter considers the Hamming distance, since experiments proved that the optimal inference attack will consider the estimated point as the one with maximum posterior probability. In both cases, they ran the adversary algorithm for 1000 times. For the optimal inference attack obtained an inference error of 0.89 km at most points for both solutions In case of Bayesian inference attack, they analyzed the number of times the algorithm made the right guess on the user's true location. In case of expected inference error, the average probability is around 50% for most points, while geo-indistinguishability shows better results. For most regions, the attack success is around 0, but there are some points where it goes up to 80%.

Another paper, "AGENT: an adaptive geo-indistinguishable mechanism for continuous location-based service" [9], introduces a novel solution to tackle the continuous GPS tracking leading to the exposure of the real location, which is a flaw of the frequently used geo-indistinguishability mechanism. Their solution uses an R-tree that stores small regions, each region containing some already sanitized locations. When the user sends a real location, their algorithm first identifies any existing sanitized locations within the given region and returns one if available. Although the number of sanitized locations is quite finite, the generated trace does look like a replica of a real trace.

3. System Architecture

In [6], we presented the architectural model of the solution, also described in Figure 1. We add a transparent layer, a browser extension that intercepts the HTML5 geolocation API, based on the implementation of LocationGuard [14]. The websites may require location information from the browser and, in case of the user's approval, it detects the exact location by querying a location service and forwards it to the website. The transparent layer that we provide intercepts the messages sent between the browser and the location service and runs an obfuscation algorithm before sending the response to the website. It is addressed mainly to mobile phones and it requires storage space, since it uses predefined maps of places where the user is located. Its aim is to protect the user's privacy against malicious or curious websites, which may track locations and get behavior patterns out of data. Since websites may remain open in browsers after usage, it is possible for them to get location data from the users without their knowledge. A browser extension is a lightweight solution that does not require additional changes, nor user technical knowledge.

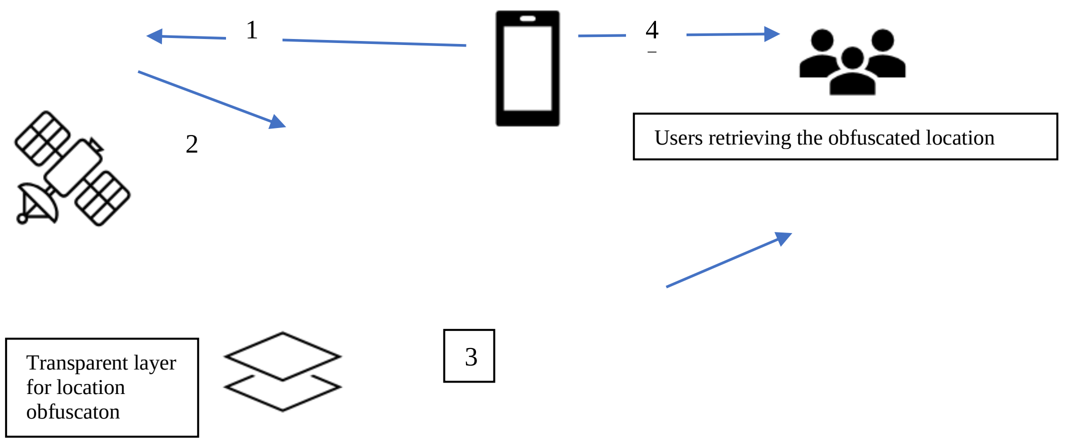
1
4
2
Users retrieving the obfuscated location
Transparent layer for location obfuscaton
3

Fig. 1 Solution architecture

4. Implementation

A fundamental piece of data is the matrix of cell popularity scores within the grid that lays over Beijing. Each cell of the grid holds a popularity score ranging from 0 to 9. The higher the score, the higher the popularity of an area. The algorithm is more likely to generate a fake location in a more popular area, using a random distribution in order to avoid selecting the same most popular area every time. A configuration file allows the user to set preferences regarding the approximate distance between the real and the false location.

The extension of the algorithm from [6] is a code section that performs the logic of determining where a new fabricated location could be placed. The first step is to read the current real location and identify the cell within the matrix where those coordinates belong. This knowledge, together with the user preferences regarding the level of obfuscation, helps contour a smaller or larger area surrounding the real location. This area contains multiple cells with individual popularity scores. Each cell is a candidate to become the cell of the newly fabricated location. If the algorithm is not at its first run, there will be a previous fake location that it generated based on the last known real location. At this point, the code attempts to identify the cell of that fake location. This can either succeed or fail. If it fails, it means that there was no previous fake location set, so the entire area determined earlier is valid. If it succeeds, however, it will determine new boundaries for the new fake location.

As explained visually in Figure 2, when a real location is being processed and it is close to a fake location, these two locations help identify the orientation of the user. The yellow marker with the label "1" has an entire area (drawn in this figure as a circle) available to generate a location. It generates the corresponding red “1”. The user moves to where the yellow “2” marker is on the map, so the algorithm attempts to generate a new fake location. However, since location “2”is more to the South-East from the previous fake location “1”, it is an indicator that the user is likely to travel in that direction. The algorithm cuts the circle, restricting the area above (to the North) and to the left (West) of the fake location “1”. This way, it can only generate a location in the desired direction, using the same location generation logic applied on smaller locations pool. The following real location is the yellow “3”. It is placed at a considerable distance away from the previous real location, yellow “2”. Because the areas of the two last real locations do not overlap, it is guaranteed that the new fake location will follow the direction of travel of the user. As a positive side effect, the distances between consecutive fake locations are proportional to the distances between their corresponding real locations, giving more plausibility to the fake trail.

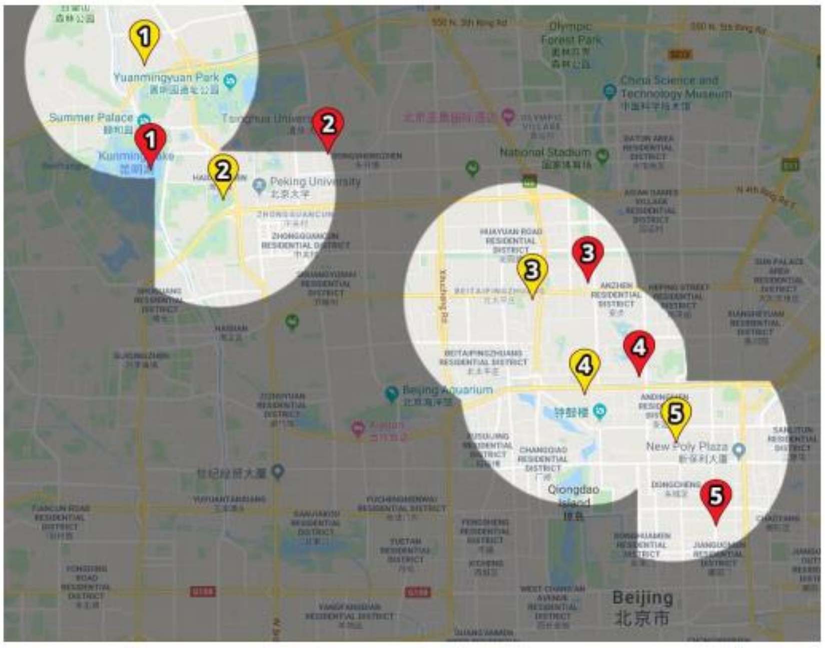

Fig. 2 Location generation limits to ensure a continuous path

Figure 3 shows all three sets of locations. With yellow markers, we represented the real trace in this scenario. The numbers on each marker show the order in which the markers were generated. The blue markers show the output of the stationary version of the algorithm from [6]. According to this result, an attacker would find out that the user has supposedly travelled West from position 1 and then South-East a considerable amount. Depending on the time frame available, it may lead to the user losing credibility in front of the attacker. The red marks the output of the algorithm described in this paper. At a first glimpse, it appears that the fifth location is missing from the result set. In fact, there are five locations but the third and fourth ones are merged. It is immediately visible that the real trace and the “red” trace are very similar regarding the heading, which is the goal of this algorithm. Each one of the “red” locations is placed on a relatively popular cell, therefore the location is plausible. An attacker without prior knowledge about the victim is not able to distinguish this fabricated trace from a real trace. For a better representation of the real and fabricated headings, a new simulation is presented in Figure 4. The false route, drawn with red arrows, follows the direction of the real route, black, while maintaining a good average distance between pairs of real and fake locations.

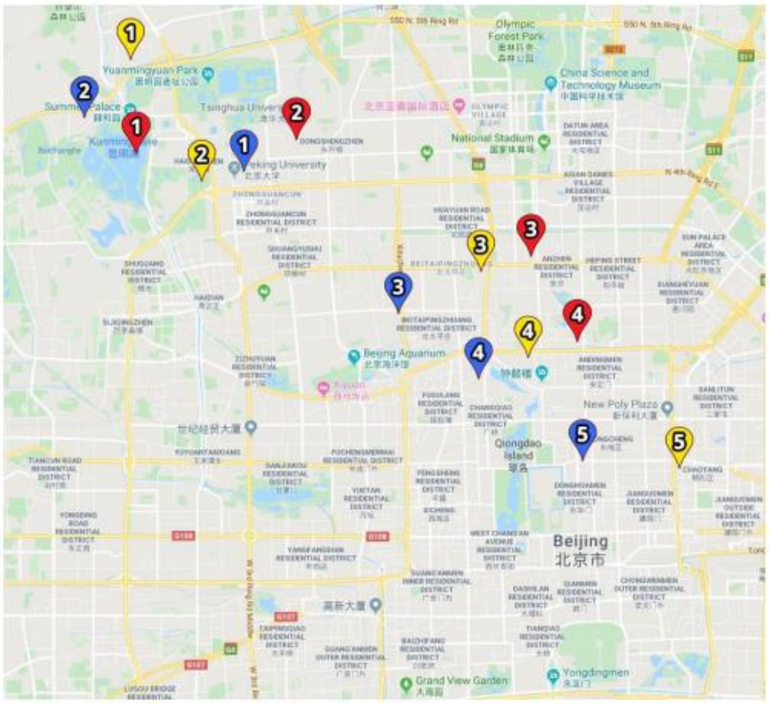

Fig.3 Comparison of the current algorithm with the stationary version and the real path

The implemented algorithm has a complexity of O(n*m), where n and m are the number of rows and columns of the location matrix. This is built based on GPS trajectories of people in GeoLife GPS Trajectories dataset [15]. Starting from the information in the plots, we can obtain the most popular areas and load the annotated maps in the transparent layer. In case of low-memory systems, the preprocessing must be done beforehand, to avoid extra load. A map file contains information on the GPS coordinates of each area, together with its popularity score.

4. Experiment design

The obfuscation layer was implemented as addOn for HTML5 API on the mobile device, with [14] as starting point. It runs on browsers that permits extensions, such as Firefox on Android or iOS devices. There are available Maps APIs, such as [15], that provide developers methods to get the exact location in browser. In case of mobile phones, which is considered the main use case, the storage space is not an issue. This type of solution may need several MB, depending on the maps that the user is willing to download. It can be compared with offline maps application, but without the visual and interactive part of the application. To test the algorithm, we created a scenario of a person travelling across Beijing, leaving a trail of five different real locations. These locations serve as input for both the last and the current generations of this algorithm. The older one treats the input as an array of five different locations and runs in a stateless manner. The result is an array of five false locations, each corresponding to one of the real locations. The newer algorithm treats the array as a series of locations, and each run remembers the output of the last run, making in run in a stateful manner. The result is also an array of five locations but distributed differently on the map.

Although this solution provides a sort of zig-zag line that wraps around the real trajectory, which is what paper [10] presents as well with their solution, this algorithm is also able to output the same location for multiple successive inputs, as explained in the previous paragraph, regarding the missing fifth red map marker. We consider this an important mechanism to enhance privacy, because from an attacker perspective the target is stationary in a credible spot on the map, while the target is in fact moving privately within an area. This mechanism also proves that the output of the algorithm is not necessarily dictated by the variations in the input data.

As a supplementary privacy feature, the algorithm avoids generating false locations on top of real locations. If the user is in a popular place in the city, the fake location could have been generated exactly in that particular spot, because it is marked as a popular place. But by doing so, an attacker that believes the output to be true, will in fact know the user's real location.

When comparing our solution with the original paper based on Laplace distribution [16], we prove a higher level of security. Their path result after running "N-Rand" essentially states that the average distance between pairs of real and fake locations is approximately 25 meters, with a standard deviation of up to 10 meters. It appears that the output route follows the real route closely, which leads to possibly identifying the real route easily, only based on the observations. This flaw may be reduced by increasing the range of the "N-Rand" algorithm. Another flaw of the Random algorithms is triangulation and presents itself when the algorithm runs multiple times on a stationary input. Multiple random output locations tend to be distributed evenly around the real location, forming a circle.

At the center of this circle is the real location, the one that is supposed to be hidden. The risk of triangulation is mitigated with an approach based on popular locations, since the output is no longer directly relative to the real location.

Our algorithm provides higher numbers related to overhead and latency than a basic algorithm based on random function distribution, such as the solution in [16]. A planar Laplace function is faster and it does not require any extra files or memory, but further experiments will expose its vulnerabilities. On the other hand, our solution will require storage space, which will not affect mobile devices, such as smartphones and processing power. The latency will go up to several seconds, depending on the accuracy level, but this should not drastically affect the user.

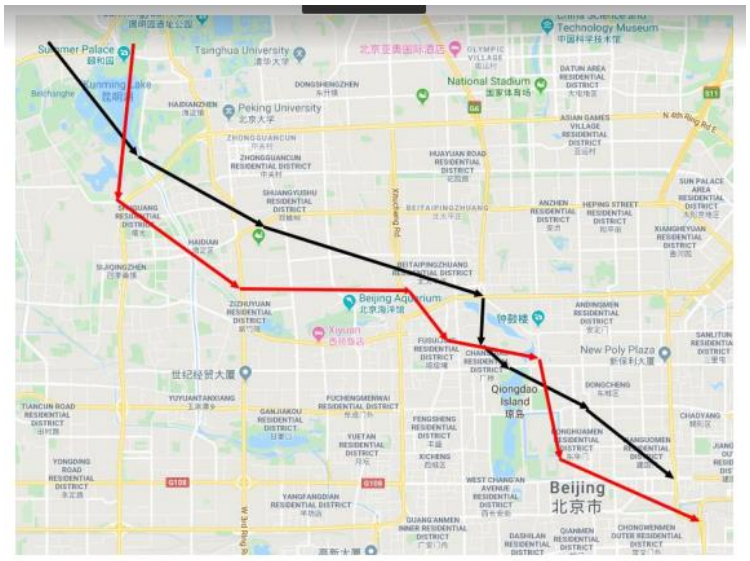
Fig.4 Real (black) and fabricated (red) path across Beijing

5. Evaluation

The first attack that the algorithm is tested against is the same-origin attack. A random input location is selected. This location is stationary and it is going to be the origin of the obfuscation requests. In the next phase, a fabricated location will be generated 1000 times, using the same input. Although paper [4] states that less runs are necessary in order to leak a real location, the result is going to be more relevant and visible when more iterations are performed.

In order to compare the results between our algorithm and the starting solution based on Laplace distribution [16], we ran the algorithm for both cases. In case of Laplace based solution, the result in presented in Figure 5. The blue marker icon in the center of the image represents the location of origin. All the blue, yellow and red clusters, together with the independent yellow markers, are locations that were generated by the algorithm, based on the input origin. The number on each cluster represents the number of generated locations that are wrapped by that particular cluster. It is immediately noticeable that the same-origin attack is effective against this obfuscation solution. After multiple runs, the majority of the output locations are distributed close to the real location, becoming gradually sparser towards the ends of the available area. The real location is always where the density of generated locations is higher.

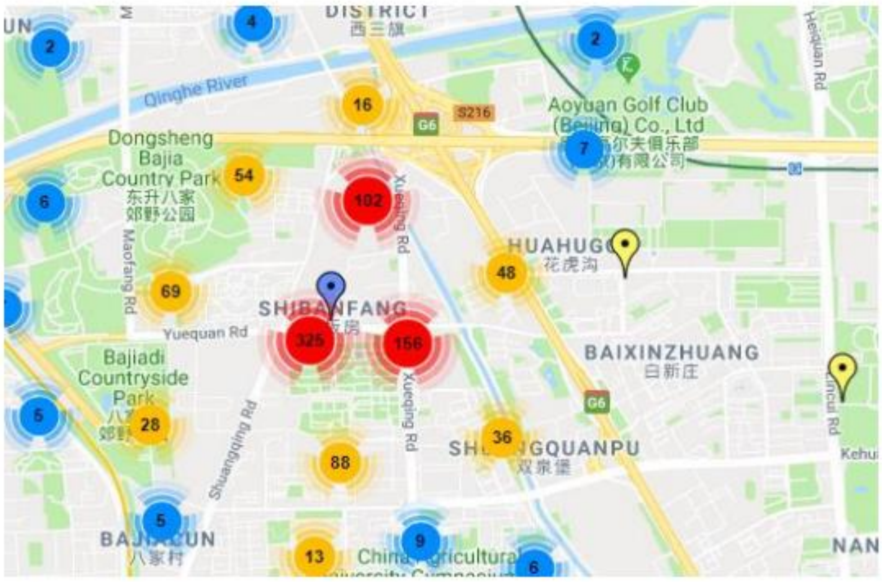

Fig. 5 Same-origin attack against Planar Laplace algorithm

In our case, the results are presented in Figure 6. The clusters are scattered across the city, with no relationship between the position of the cluster relative to the origin and the density of the cluster. There are larger clusters in the bottom-left quadrant because that is where some of the most popular areas are. As a general rule, the density of a cluster is only proportional to the popularity of that area of the map. As a precautionary measure, if the real location of a user happens to be in such a popular place, no false location will be generated in that place. The output locations will only be generated in popular places different than the origin. The algorithm is not vulnerable to the same-origin attack, because the distribution of the generated locations on the map does not point towards the original location.

The second type of attack is described as testing the algorithm with multiple different inputs, hoping to identify the input that generates one specific output. In general, the attacks of this fashion are called brute-force attacks. This type of attack is mainly effective in the case of deterministic algorithms, but to some extent can be used to guess approximate input areas when deployed against location obfuscation algorithms. The attacker starts with a given output (obfuscated) location. For instance, we can assume that this output location is placed at the North end of a fairly large city. The attacker feeds the algorithm with different input locations, attempting to generate various outputs. While he may not be able to choose between multiple apparently valid inputs, he will most likely notice that inputs close to the South end of the city do not generate outputs in the North (because it is too far). Based on this observation, he is able to place the real location of the user somewhere towards the North, or the city center, or slightly towards North-West or North-East. For a given fixed output, we will run the algorithm with various inputs and analyze the results.

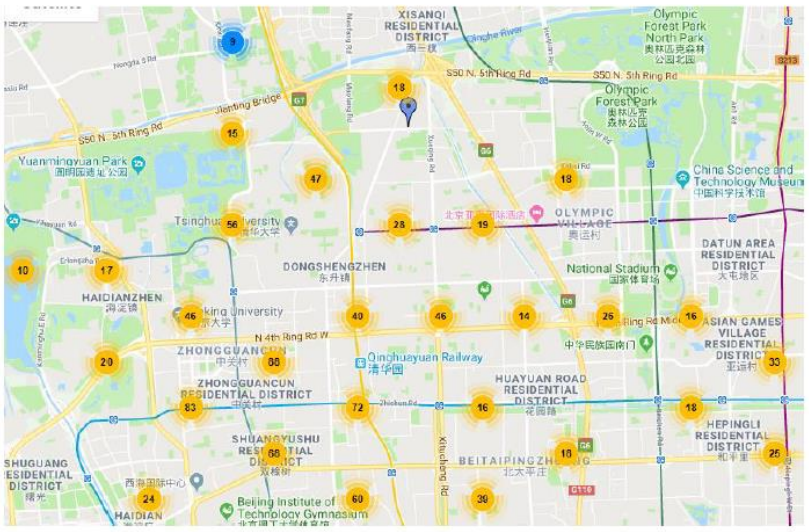

Fig. 6 Same-origin attack against our location obfuscation solution

The behavior of the algorithm varies considerably with respect to the position of the designated output. It scans a limited area around the input location and chooses a random map cell that is more popular than a given popularity threshold. Naturally, if a map cell does not have a high enough popularity score, it will never be chosen as output. If the designated output falls in any one of these cells, this attack fails because there will be no input that could generate the expected output. For the edge case where the output location is placed in the only highly popular spot in a region, the algorithm is expected to return that map cell very often.

In a regular scenario, the map is likely to contain multiple popular locations in the fairly wide area surrounding the designated output. A regular scenario was tested and the result was plotted in Fig. 7. This bar graph was generated with data collected based on the same map and input location that were used for the rest of the experiment. The score of the map cell where the input belongs was changed with each iteration, while all the other scores remained the same. The popularity scores are normalized to fit in the range from 0 to 9, 0 being an area with no interest from pedestrians and 9 being a highly popular area for many citizens. When the popularity score of the input location area is lower than a certain threshold, currently set to 3, there are no locations generated there. Then, as the score increases, some of the generated locations are placed next to the input location. Since the scores map is fairly balanced, even with a maximum score of 9 points, only 2.4% of locations are placed in the observed area. This balance is enforced by the rule that even if an area has the maximum score, it is not guaranteed to be selected. Instead, it only has a slightly higher chance (weight, in a weighted random distribution) compared to other areas with scores of 8 or lower.

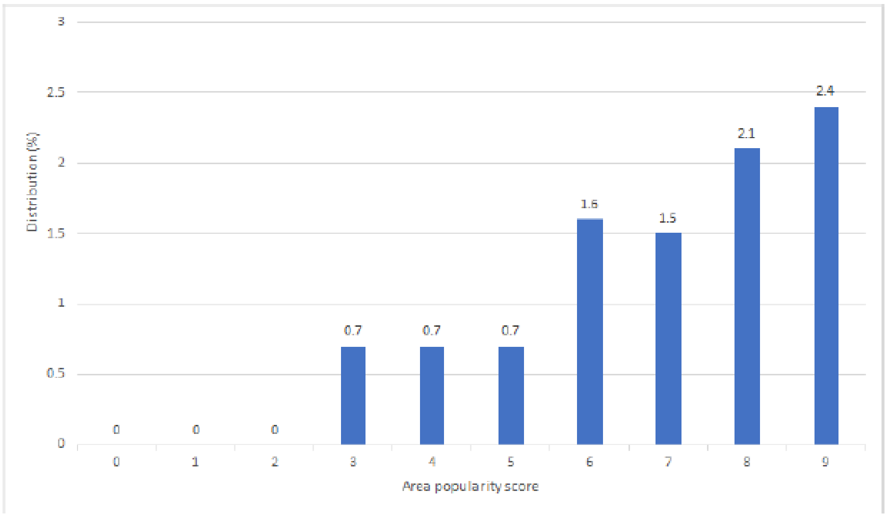

Fig.7 Location distribution based on a fixed output

The experimental result is shown in Figure 8. Given the popularity score of the map cell where the blue marker is, only some inputs from the North-East side were able to generate the expected output. We also made the assumption that the attacker does not know that the algorithm uses a grid and rectangular shapes. Therefore, the attacker is now able to draw a circle with the radius equal to the distance between the designated output and the farthest good input. With enough iterations of the brute-force attack, it is plausible that the attacker may be able to detect a fairly accurate radius for the green circle. Same as before, the real location is somewhere within this green circle, but there are no further clues as to where precisely it may be. As the obfuscation radius increases, the green circle's radius will grow as well, and the privacy level will be higher.

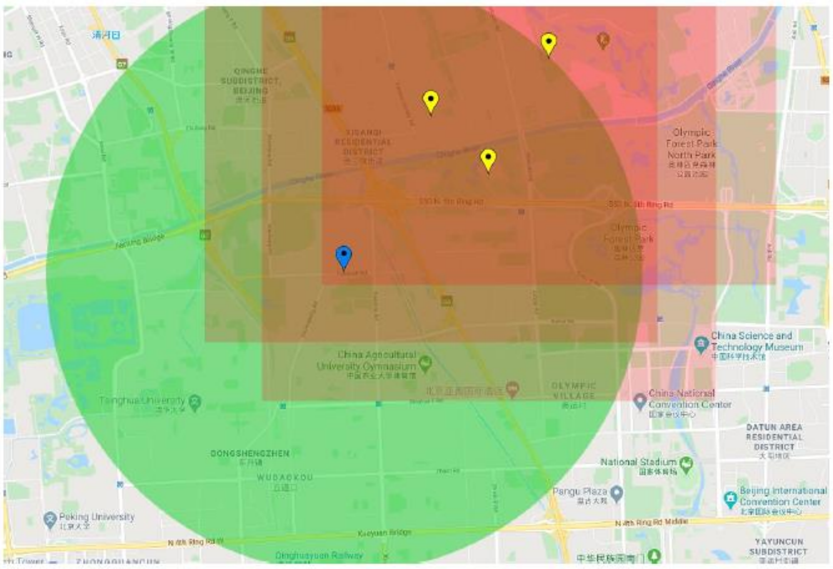
Fig. 8 Brute force attack against the location obfuscation algorithm

6. Conclusions

To sum up, we started from the implementation of the stationary algorithm in [6], expanding a stateless solution to a stateful version, that takes into consideration the previous locations of the user. We generate new locations based on the popularity score of the points of interest nearby the user and on his/her pervious location. The results are consistent with the theoretical expectations. Given a trajectory based on real locations, the output appears very similar in orientation, with locations generated in plausible places.

After building the algorithm, we tested it against state-of-the art attacks over location privacy obfuscators. The algorithm behaves similarly in most cases, but there are a few edge scenarios to be taken into account. When the obfuscated location, somehow, is placed in an unpopular place on the map, the algorithm will never be broken by the brute- force attack. When the area of the obfuscated location is minimally popular and some more popular areas are in the vicinity, the algorithm is likely to choose those popular places as output. This may lead the attacker into believing that the area containing the real location is smaller than it actually is. However, the usual scenario matches the behavior of the original algorithm, meaning that the improved algorithm is also offering protection against the brute-force attack.

Regarding the idea of extending the solution to embedded system, the work in [17, 18] has proved that the a properly implemented secure communication channel requires only a small fraction of the energy. In this case, a low complexity algorithm would not affect IoT the devices, especially if there are only a few necessary preprocessed maps.

R E F E R E N C E S

[1] S. Zeng, Y. Mu, M. He, M. and Y. Chen, “New approach for privacy-aware location-based service communications” in Wireless Personal Communications, 101(2), 2018, pp. 1057-1073.

[2] Z. Li, Q. Pei, I. Markwood, Y. Liu, M. Pan and H. Li, “Location privacy violation via GPS-agnostic smart phone car tracking” in IEEE Transactions on Vehicular Technology, 67(6), 2018, pp. 5042-5053.

[3] Z. Wu, G. Li, Q. Liu, G Xu and E. Chen, “Covering the sensitive subjects to protect personal privacy in personalized recommendation” in IEEE Transactions on Services Computing, 11(3), 2016, pp. 493-506.

[4] L. Yu, L. Liu and C. Pu, "Dynamic Differential Location Privacy with Personalized Error Bounds", 2017.

[5] K. Fawaz and K.G. Shin, “Location privacy protection for smartphone users” in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 239-250.

[6] I. M. Florea, D. Vornicu, J.A. Văduva and R. Rughiniș, „TEACHING PRIVACY THROUGH THE DEVELOPMENT AND TESTING OF A LOCATION OBFUSCATION SOLUTION” In The International Scientific Conference eLearning and Software for Education, vol. 2, pp. 145-153, 2020.

[7] N. Bordenabe, K. Chatzikokolakis and C. Palamidessi, "Optimal Geo-Indistinguishable Mechanisms for Location Privacy" in In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. 2014. p. 251-262

[8] G. Theodorakopoulos, "The Same-Origin Attack against Location Privacy," in Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society, Denver, 2015.

[9] P. Wightman, W. Coronell, D. Jabba, M. Jimeno and M. Labrador, "Evaluation of Location Obfuscation Techniques for Privacy” in Location Based Information Systems" in 2011 IEEE Third Latin-American Conference on Communications. IEEE, 2011. p. 1-6.

[10] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[11] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, J.-P. Hubaux, “Quantifying location privacy” in Security and Privacy (SP), 2011 IEEE Symposium on, pages 247–262, May 2011.

[12] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, “Protecting location privacy: Optimal strategy againstlocalization attacks” in Proceedings of ACM CCS, pages 617–627, NewYork, USA, 2012.

[13] X. Ma, J. Ma, H. Li, Q. Jiang and S. Gao, "AGENT: an adaptive geo-indistinguishablemechanism for continuous location-based service," in *Peer-to-Peer Networking and Applications*, *11*(3), pp. 473-485 2018.

[14] Website: https://github.com/chatziko/location-guard; Last accessed: October 2021

[15] Website: https://download.microsoft.com/download/F/4/8/F4894AA5-FDBC-481E-9285-D5F8C4C4F039/Geolife%20Trajectories%201.3.zip; Last accessed: October 2021

[16] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis & C. Palamidessi “Geo-indistinguishability: Differential privacy for location-based systems” In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 901-914), 2013

[17] R. Tataroiu, F.A. Stancu & D.C. Tranca, “Energy considerations regarding transport layer security in wireless iot devices” In 22nd International Conference on Control Systems and Computer Science (CSCS) , pp. 337-341, 2019.

[18] D.C. Tranca, D. Rosner & A.V. Palacean, Autonomous flexible low power industrial IoT controller for solar panels cleaning systems, In 2017 21st International Conference on Control Systems and Computer Science (CSCS), pp. 106-112, 2017