De-anonymization attack on geolocated datasets

Sébastien Gambs, Marc-Olivier Killijian, Miguel Nuñez del Prado Cortez

De-anonymization attack on geolocated datasets

Sébastien Gambs
Univ. de Rennes 1 - INRIA / IRISA
Campus Universitaire de Beaulieu, 35042 Rennes, France
sgambs@irisa.fr

Marc-Olivier Killijian
LAAS - CNRS
7 avenue du Colonel Roche
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS
F-31077 Toulouse, France
marco.killijian@laas.fr

Miguel Núñez del Prado Cortez
LAAS-CNRS
7 avenue du Colonel Roche
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS
F-31077 Toulouse, France
mnunezde@laas.fr

ABSTRACT

With the advent of GPS-equipped devices, more and more geolocated datasets are being collected everyday, thus raising the issue of the privacy risks incurred by the individuals whose movements are recorded. In this work, we focus on a specific inference attack called the de-anonymization attack, by which an adversary tries to infer the identity of a particular individual behind a mobility trace. More specifically, we propose an implementation based on a mobility model called Mobility Markov Chain (MMC). A MMC is built out from the mobility traces observed during the training phase and is used to perform the attack during the testing phase. We design distance metrics between MMCs and combine these distances to build de-anonymizers that can re-identify users in an anonymous dataset. Moreover, experimentations conducted on real datasets show that the attack is efficient in terms of accuracy and resilient to sanitization mechanisms.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

1. INTRODUCTION

Recently, with the advent of ubiquitous devices and smartphones equipped with positioning capacities such as GPS, more and more mobility traces are collected and gathered in the form of geolocated datasets by cellphone companies or research institutions. Some of these geolocated datasets are made available in public repositories and can be used for research or for industrial purposes (*e.g.*, to optimize the placement of cellular towers, to conduct market or sociological studies or to analyze the flow of traffic inside a city). At the same time, these datasets are composed of the mobility traces of hundred or thousands of individuals [**?**, **?**, **?**, **?**], thus raising the issue of the privacy risks incurred by these individuals. For instance, from the movements of an individual it is possible to infer his points of interests (such as his home and place of work) [**?**, **?**, **?**, **?**], to predict his past, current and future locations [**?**, **?**], or even to infer his social network [**?**].

In this work, we are interested in a particular form of inference attack called *de-anonymization attack*, by which an adversary tries to infer the identity of a particular individual behind a mobility trace. More precisely, we suppose that the adversary has been able to observe the movements of some individuals during a non-negligible amount of time (*e.g.*, several days or weeks) in the past during the training phase. Later, the adversary has access to another geolocated dataset that contains some of the individuals observed during the training phase. The objective of the adversary is then to de-anonymize this new dataset (called the testing dataset) by linking it to the corresponding individuals of the training dataset. Simply replacing the real names of individuals by pseudonyms before releasing a dataset is usually not sufficient to preserve the anonymity of their identities because the mobility traces themselves contain information that can be uniquely linked to an individual. In addition, a dataset can be sanitized before being released, by adding spatial and temporal noise, but nevertheless the risk of re-identification through de-anonymization attack is still possible.

In this paper, we propose a new method to de-anonymize geolocated data based on a mobility model called Mobility Markov Chain (MMC) [**?**]. A MMC is a probabilistic automaton, in which each state corresponds to a point of interest (or several points of interests) characterizing the mobility of an individual and an edge indicates a probabilistic transition between two states (*i.e.*, points of interests). Each node can potentially have a semantic label attached to it such that home, work, leisure, sport, ... A MMC is built out of the mobility traces observed during the training phase and is used as a tool to perform the de-anonymization attack during the testing phase. More precisely, the mobility of each individual both from the training and testing sets is represented in the form of a MMC. Afterwards, a distance measure is computed between MMCs from the training and testing sets in order to identify the closest individuals in terms of mobility. In short, the gist of this method is that the mobility of an individual can act as a signature, thus playing the role of a quasi-identifier. Therefore, if the adversary knows Alice and a signature of her mobility (*e.g.*, he has learnt her MMC from the training set), he can try to identify her by finding a signature in the testing set that

matches her signature.

The outline of the paper is the following. First, in Section **??**, we review some related work on de-anonymization attack and mobility models. In section **??**, we briefly introduce the background on Mobility Markov Chains necessary to the understanding of our work. Afterwards in Section **??**, we introduce the distance measures between MMCs that we have designed to be able to quantify the closeness between two mobility behaviors, while in Section **??** we present how to build predictors (which we call *de-anonymizers*) based on distance measures to efficiently and accurately de-anonymize geolocated data. Finally, we evaluate experimentally the efficiency of the proposed attack on real geolocated datasets in Section **??** before concluding in Section **??**.

2. RELATED WORK

An *inference attack* corresponds to any process by which an adversary that has access to some data related to individuals (plus possibly some auxiliary information) tries to deduce new personal information that was not explicitly present in the original data. For instance, a famous inference attack was conducted by Narayanan and Shmatikov on the "Netflix dataset" [**?**]. This dataset is a sparse high dimensional data containing ratings on movies of more than 500 000 subscribers from Netflix that was supposed to have been anonymized before its release for the Netflix competition[1]. However, Narayanan and Shmatikov have performed a de-anonymization attack that was able to successfully re-identify more than 80% of the Netflix subscribers by using the Internet Movie Database[2] (IMDB), another database of movie ratings, as supplementary background knowledge.

Inference attacks have also been developed specifically for the geolocated context. For instance, Mulder *et al.* [**?**] have proposed two methods to profile users in a GSM network that can also be used to perform a de-anonymization attack. The first method is based on constructing a Markovian model of the mobility behavior of an individual while the second considered only the sequence of cell IDs visited. The first method is relatively similar to our work with two major differences due to the fact their method is based on a cellular network. First, they use the static GSM cells as the states of the Markovian model, while we dynamically learn the Points Of Interest (POIs). Therefore, in our setting two individuals do not necessarily have any POIs in common, whereas with GSM cells, individuals living in the same area have a high probability of sharing some POIs (*i.e.*, cells). As a consequence, the second main difference is that in their work the transitions are only between neighboring GSM cells, as it is impossible to "jump" from one cell to another if they are not adjacent. Their attack was validated through experimentations using cell locations from a MIT Media Lab dataset[3] [**?**] that includes information such as call logs, Bluetooth devices in proximity, cell tower IDs, application usage and phone status.

Finally in the work of Mulder *et al.*, there was no difference between the training and testing test, which actually consist both in the complete dataset. This overlap between the training and testing set induces a bias on the obtained results. The authors have also used an agglomerative hierarchical clustering algorithm to group users according to a similarity measure called the cosine similarity. They observed that if the currently profiled user shares the same cluster with other users, there is a high risk of making a mistake on the person when performing the de-anonymization attack, thus leading to a high rate of false positives.

Zang and Bolot [**?**] have performed a study of the top n most frequently visited places by an individual in a GSM network and show how they can act as quasi-identifiers to re-identify anonymous users. Their study was performed on the Call Data Record (CDR) from a nationwide US cellular provider that was collected over a month and contains approximately 20 millions users. From this dataset, the authors have identified the top n most frequent places for each user at different levels of granularity for space (sector, cell, zip code, city, state and country) and time (day and month). Their inference attack was able to successfully re-identify 35% of the population studied when the adversary has no background knowledge and even up to 50% when the adversary can use the knowledge of the social network of users as auxiliary information. The social network was constructed by creating a social relationship between two individuals that have called each other at least once in the past. In their analysis, the authors emphasize that the distance between home and work can be an indicator of the privacy level for an individual. In particular, the larger this distance, the higher the risk that this individual can be de-anonymized.

Ma *et al.* [**?**] proposed an inference attack to de-anonymize users in a geolocated dataset along with a metric to quantify the privacy loss of an individual. Two datasets were used in this study, one stored in the Crawdad repository recording the movements of San Francisco YellowCabs [**?**] and the other about Shanghai city public buses[4]. Two types of adversary models were considered: the *passive* one, collecting the whereabouts of individuals from a public source (possibly sanitized) and the *active* one that can deliberately participate or perturb the data collection in order to gain additional knowledge about the location of some specific individuals. To retrieve the identities of individuals, the authors imagine four different estimators that the adversary can use to measure the similarity between mobility traces (for instance between the original traces and the sanitized ones). The *Maximum Likelihood Estimator* (MLE) relies on the Euclidean distance to compute the similarity between mobility traces. The *Minimum Square Approach* (MSQ) computes the negative sum of the square of the absolute value of the difference between the original traces and the sanitized ones. The *Basic Approach* (BAS) assumes that the traces have been perturbed by uniform noise. Therefore, a mobility trace will be considered to be similar to another trace if it is contained within a sphere of radius r centered on the original trace. Finally, the *Weighted Exponential Approach* (EXP) is similar to BAS, with the exception that no assumption is made on the type of noise generated or the knowledge of the adversary about this noise.

The methods proposed by Ma *et al.* perform particularly well, approaching a success rate of de-anonymization of 80% to 90%, and this even in the presence of noise. However, much like Mulder *et al.* [**?**], the authors did not really split the geolocated dataset between a training set and a testing set. More precisely, they assume that both type of adver-

[1] `http://www.netflixprize.com`

[2] `http://www.imdb.com`

[3] `http://reality.media.mit.edu/dataset.php`

[4] To the best of our knowledge, this dataset is not publicly available contrary to the other one.

saries (*i.e.*, passive and active ones) can pick up the information they used to build their mobility model of an individual directly from the testing set (*i.e.*, the same dataset on which the success of the de-anonymization attack is tested). More precisely, the first version of the passive adversary (called *A1*) picks up his data directly into the test set while the second version of the passive one (*A2*) relies on data that have been generated on data generated at the same time at the test data although not directly present in this dataset. Basically, the method proposed by Ma *et al.* extracts the best signature of the mobility of an individual and evaluate, depending on the size of this signature (ranging from 1 to 30 timestamped positions), how it uniquely identifies the target individual.

3. MOBILITY MARKOV CHAIN

A *Mobility Markov Chain* (MMC) [**?**] models the mobility behavior of an individual as a discrete stochastic process in which the probability of moving to a state (*i.e*, point of interests) depends only on the previously visited state and the probability distribution on the transitions between states. More precisely, a MMC is composed of:

- A set of states $P = \{p_1, \dots p_n\}$, in which each state is a frequent POI (ranked by decreasing order of importance), with the exception of the last state p_n that corresponds to the set made of all infrequented POIs. POIs are usually learned by running a clustering algorithm on the mobility traces of an individual. These states generally have an intrinsic semantic meaning and therefore semantic labels such as "home" or "work" can often be inferred and attached to them.

- Transitions, such as $t_{i,j}$, represent the probability of moving from state p_i to state p_j. A transition from one state to itself is possible if the individual has a non-null probability from moving from one state to an occasional location before coming back to this state. For instance, an individual can leave home to go to the pharmacy and then come back to his home. In this example, it is likely that the pharmacy will not be extracted as a POI by the clustering algorithm, unless the individual visits this place on a regular basis.

Note that many mobility models relying on a Markov chains have been proposed in the past [**?**, **?**], including the use of hidden Markov models for performing inference attacks [**?**]. In a nutshell, building a MMC is a two step process. During the first phase, a clustering algorithm is run to extract the POIs from the mobility traces. In the study of Gambs *et al.* [**?**], a clustering algorithm called Density Joinable cluster (DJ-Cluster) was used that we also rely on in our work, but of course other clustering algorithms are possible. In the second phase, the transitions between those POIs are computed.

DJ-Cluster takes as input a trail of mobility traces and 3 parameters: the minimal number of points $MinPts$ needed to create a cluster, the maximum radius r of the circle within which the points of a cluster should be contained and a distance d at which neighboring clusters are merged into a single one. DJ Cluster works in three phases. During the first phase, which corresponds to a pre-processing step, all the mobility traces in which the individual is in movement (*i.e.*, whose speed is above some small predefined value) as well as subsequent static redundant traces are removed. As a result, only static traces are kept. The second step consists in the clustering itself: all remaining traces are processed in order to extract clusters that have at least $MinPts$ points within a radius r of the center of the cluster. Finally, the last phase merges all clusters that have a trace in common or whose centroids are within d distance of each other.

Once the POIs (*i.e.*, the states of the Markov chain) are identified, the probabilities of the transitions between states can be computed. To realize this, the trail of mobility traces is examined by chronological order and each mobility trace is tagged with a label that is either the number identifying a particular state of the MMC or the value "unknown". Finally, when all the mobility traces have been labeled, the transitions between states are counted and normalized by the total number of transitions in order to obtain the probabilities of each transition. Note that MMC can be either represented as a transition matrix or in the form of a graph in which nodes correspond to states and arrows represent the transitions between along with their associated probability. When the MMC is represented as a transition matrix of size $n \times n$, the rows and columns correspond to states of the MMC while the value of each cell is the probability of the associated transition between the corresponding states.

4. DISTANCES BETWEEN MOBILITY MARKOV CHAINS

In this section, we propose four different distance metrics between mobility Markov chains that quantify the similarity between two mobility behaviors. These distance metrics are based on different characteristics of the MMCs and thus give different but complementary results. We will rely on these distance metrics in the next sections to perform the de-anonymization attack.

4.1 Stationary distance

The intuition behind the *stationary distance* is that the distance between two MMCs corresponds to the sum of the distances between the closest states of both MMCs. In order to compute the stationary distance, the states of the MMCs are paired in order to minimize this distance. As a result, it is possible that a state from the first MMC can be paired with several states of the second MMC (this is especially true if the MMCs are of different size).

The computation of the stationary distance is based on the stationary vectors of the MMCs. The stationary vector is a column vector obtained by multiplying the MMC transition matrix by itself repeatedly until convergence. The stationary distance is directly computed from the stationary vectors of two MMCs (hence its name). More precisely, given two MMCs, M_1 and M_2, the stationary vectors, respectively V_1 and V_2, of each model are computed. Afterwards, Algorithm **??** is run on these two stationary vectors. For each state in V_1 (line 2), the algorithm searches for the closest state in V_2 (lines 5 to 11) and then multiply the distance between these two states by the corresponding probability of the stationary vector of the state of V_1 currently considered (line 12). Once the algorithm has taken into account all states from V_1, the current value computed represents the distance from M_1 to M_2 ($distance_{AB}$ in line 1, Algorithm **??**). This distance is not symmetric as such and therefore in order to symmetrize it, Algorithm **??** is called once again but on V_2 and V_1 in

order to obtain the distance from M_2 to M_1 ($distance_{BA}$ of line 2, Algorithm **??**). The result is made symmetrical by computing the average of these two distances (line 3, Algorithm **??**).

Algorithm 1 Stationary_distance(V_1, V_2)

```
1: distance = 0
2: for i = 1 to n_1 (the number of nodes in V_1) do
3:    MinDistance = 1000000 kilometers
4:    Let p_i be the i^th node of V_1
5:    for j = 1 to n_2 (the number of nodes in V_2) do
6:       Let p_j be the j^th node of V_2
7:       CurrentDistance = Euclidean_Distance(p_i, p_j)
8:       if (CurrentDistance < MinDistance) then
9:          MinDistance = CurrentDistance
10:      end if
11:   end for
12:   distance = distance + Prob_{V_1}(p_i) × MinDistance
13: end for
14: return distance
```

Algorithm 2 Symmetric_stationary_distance(V_1, V_2)

```
1: distance_AB = Stationary_distance(V_1, V_2)
2: distance_BA = Stationary_distance(V_2, V_1)
3: distance = (distance_AB + distance_AB)/2
4: return distance
```

4.2 Matching distance

The *matching distance* is similar to the stationary distance in the sense that this distance also corresponds to the sum of distances between the states of these two MMCs. However, the pairing of the states between the MMCs is done in a different way as each state from the first MMC is paired with one and only one state of the second MMC.

The matching distance is based on the Hungarian method [**?**], which is a polynomial-time combinatorial optimization algorithm for assignment problems. More precisely, the method works as follows. Given two MMCs, M_1 and M_2 and their corresponding stationary vectors V_1 and V_2, Algorithm **??** first verifies if the number of nodes in both models is the same. If not, we assume without loss of generality that M_2 is the MMC with the fewer number of states. Before continuing the computation of the distance, "dummy" states are added to M_2. Each dummy state is a copy of the centroid of the states of M_2. Once this process is completed, the number of states in V_1 and V_2 is the same. Afterwards, the *distance matrix* ($D_{i,j}$) and the *minimization matrix* ($Min_{i,j}$) are computed. The distance matrix contains the Euclidean distance between each pair of nodes in M_1 and M_2 (lines **??**-**??**), in which nodes in M_2 form the rows of the matrix and nodes in M_1 correspond to the columns of $D_{i,j}$. For instance, the distance $D_{i,j}$ is the Euclidean distance between the node i in M_2 and the node j in M_1. The minimization matrix is equivalent to the distance matrix, except that each distance $Min_{i,j}$ in which node i is a dummy is assigned a large default value, such as 100 000 kilometers (line **??**). In short, the goal of this large value is to guide the Hungarian method towards giving priority to real states instead of dummy ones.

The minimization matrix, which is squared, is passed as input to the Hungarian method. The Hungarian method then computes the optimal assignment between states from M_1 and M_2 that minimizes the global sum of distances between pair of states and such that each state from V_1 is exactly matched with one state of V_2 (and *vice-versa*). This optimal matching is returned as a matrix $Index$ composed of two columns. Each column contains respectively the indexes of rows and columns of the optimal assignment for the minimization matrix $Min_{i,j}$. Figure **??** illustrates graphically an optimal assignment between two MMCs (one for Alice and one for Bob). The matrix $Index$ corresponding to Figure **??** is $I = \{\{1,1\},\{2,3\},\{3,2\},\{4,3\},\{5,4\}\}$.

For each pair of matching state, the distance between these two (non dummy) states is multiplied by the average of probabilities of stationary vectors V_1 and V_2. However, when one of the state in M_2 is dummy, the nearest real state in M_2 is identified, and the distance between these two states is then multiplied by the probability corresponding to the "orphan" state in V_1 divided by two (lines **??**–**??**). The distance returned is the sum of the values computed for each matching (line **??**).

Algorithm 3 Matching_distance(V_1, V_2)

```
1: Let D be the distance matrix D and Min the minimization matrix
2: Let n_1 be the number of nodes in V_1
3: Let n_2 be the number of nodes in V_2 (we suppose that V_2 is the stationary vector with the fewest number of states if the number of states between V_1 and V_2 is different)
4: Add fake states to V_2 if necessary to ensure that n_2 = n_1
5: for i = 1 to n_1 do
6:    for j = 1 to n_2 do
7:       D_{i,j} = Euclidean_distance(p_i, p_j)
8:       if (p_i or p_j is fake) then
9:          Min_{i,j} = 100000
10:      else
11:         Min_{i,j} = D_{i,j}
12:      end if
13:   end for
14: end for
15: Index = HungarianAssignment(Min)
16: Dist = 0
17: for i = 0 to n_1 do
18:   row = Index_{i,0}
19:   col = Index_{i,1}
20:   if (V_2.node(col) is not fake) then
21:      proba = (V_1(row) + V_2(col))/2
22:      Dist = Dist + D_{row,col} × proba
23:   else
24:      proba = (V_1.stationaryVector(row) + 0)/2
25:      Dist = Dist + NearestState(V_1.node(row), V_2.nodes)
26:   end if
27: end for
28: return Dist
```

4.3 Density-based distance

Like the stationary and the matching distance, the *density-based distance* is simply the sum of the distances between pairs of MMCs states. However, the MMCs states are paired according to their rank once they are sorted using their corresponding probability in the stationary vector.

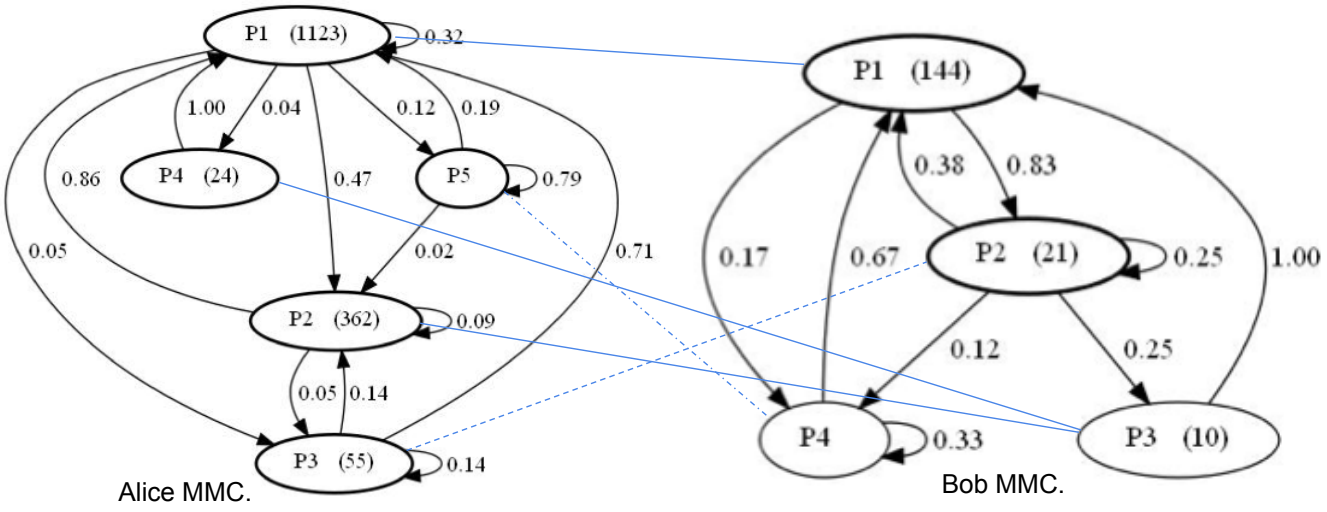

Figure 1: Example of the optimal matching of two MMCs. The number in each node corresponds to the numbers of traces that falls inside this state.

First, given two MMC models M_1 and M_2, the nodes in both models are sorted by decreasing density (Algorithm **??**). Therefore, the first node will be the one with the highest stationary probability (lines **??** and **??**). Afterwards, the first node of M_1 is matched with the first node of M_2 and the same goes for the rest of the nodes. Finally, the sum of all distances between matched nodes is computed (line **??** to **??**) and the algorithm outputs the total distance.

Algorithm 4 Densitybased_distance(V_1, V_2)

```
1: Sort the states of V1 by decreasing order of density
2: Sort the states of V2 by decreasing order of density
3: distance = 0
4: for i = 1 to min(n1, n2) do
5:    Let pa be the i^th node of V1
6:    Let pb be the i^th node of V2
7:    distance = distance + Euclidean_distance(pa, pb)
8: end for
9: return distance
```

4.4 Proximity distance

The intuition behind the *proximity distance* is that two MMCs should be considered as very close if they share "important" states. For instance, if two individuals share both their home and place of work they should be considered as being very similar. Moreover, like for the density-based distance, the importance of a state is directly proportional to the frequency at which it is visited. The first states ordered by decreasing order of importance are compared, then the second, then the third, and so on. The proximity score obtained for sharing the first states is consider twice as important as the score for sharing the second states, which is itself twice as important as the sharing of the third states, and so on.

Given two MMC models M_1 and M_2, a threshold (Δ) and a *rank*, Algorithm **??** verifies for each pair of nodes between M_1 and M_2 (line 2) if the Euclidean distance between them is less than the threshold Δ (line 6). For instance, considering the nodes of Figure **??**, the comparison is done with the nodes from POI_1 to POI_4 (the POI_5 in Alice's MMC is not taken into account as it is composed of all the infrequent POIs). If the distance between two nodes is less than Δ, the value of *rank* is added to the score value (line 7). Afterwards, *rank* is divided by two (lines 9-12). Once all the pair of nodes have been processed, the global distance is obtained by taking 1 and dividing it by the global score if this score is positive (lines 15-17). Otherwise, the distance outputted is a large value (*e.g.*, 360 000 kilometers).

Algorithm 5 Proximity_distance(V_1, V_2)

```
1: score = 0, rank = 10
2: for i = 1 to min(n1, n2) do
3:    Let pa be the i^th node of V1
4:    Let pb be the i^th node of V2
5:    distance = Euclidean_distance(pa, pb)
6:    if (distance < Δ) then
7:       score = score + rank
8:    end if
9:    rank = rank/2
10:   if (rank = 0) then
11:      rank = 1
12:   end if
13: end for
14: distance = 360000
15: if (score > 0) then
16:   distance = 1/score
17: end if
18: return distance
```

It is worth noting that both the stationary distance, the matching distance, and the density-based distance are all numerical, in the sense that they are a sum of the Euclidian distance of some pairing of the states. The proximity distance, however, is completely different as it is based on the semantics behind the MMCs. Indeed, the first state in the model is inferred as being very representative of the mobility of an individual (*e.g.*, home), the second as quite

representative (*e.g.*, the place of work), and two individuals are considered as very similar if they share these two places. Thereafter, we will see how these distance can be used to build de-anonymizers and how their diversity can be exploited and combined to enhance the success of the de-anonymization attack.

5. DE-ANONYMIZERS

In this section, we rely on the distance metrics proposed in the previous section to build statistical predictors in order to perform a de-anonymization attack. We call such a predictor, a *de-anonymizer* in reference to its main objective. A de-anonymizer takes as input the MMC representing the mobility of an individual and tries to identifies within a set of anonymous MMCs, the one that is the most similar (*i.e*, the closest in terms of distance). For example, a de-anonymizer can learn from the training set a MMC that represents the mobility of Alice and later look for the presence of Alice in the testing with the goal of identify her if this is the case. De-anonymizer can be based on one distance metric or a combination of them.

The de-anonymizers works on the distance matrix that contains the distances between different MMCs. These distances can correspond to any distance function such as $\mathsf{Dist} \in \{Stationary, Matching, Proximity, DensityBased\}$. For instance, in the distance matrix the rows can represent Alice, Bob and Charlie, whose MMCs have been learnt from the training dataset, while the columns correspond to anonymous MMCs (*e.g.*, uMMC1, uMMC2, uMMC3 and uMMC4) learnt from the testing dataset that we aim to re-identify. Therefore, the cells of the matrix contain the distance between the known models and the unknown ones as illustrated in Table **??**.

	$uMMC_1$	$uMMC_2$	$uMMC_3$	$uMMC_4$
Alice	10	40	30	20
Bob	13	27	17	3
Charlie	45	22	7	22

Table 1: Illustration of a distance matrix between MMCs.

Several de-anonymizers were designed that works directly on the distance matrix:

- The *minimal-distance* de-anonymizer considers that in each row, the MMC with the minimal distance (*i.e.*, the column) is the individual corresponding to the identity of the row. For instance in the above example, this de-anonymizer would consider that the individual behind uMMC1 is Alice.

- The *minimal-value* de-anonymizer takes as input two different distance matrices (of size $m \times n$) based on different distances Dist. These original distance matrices are transformed into another one (also of size $m \times n$) that contains only the minimal values from the original matrices. For example, starting from two distance matrices M_1 and M_2, the resulting transformed matrix $\hat{M}$ is such that $\hat{M}(i,j) = \mathsf{Min}(M_1(i,j), M_2(i,j))$. Once this transformed matrix $\hat{M}$ is built, the minimal distance de-anonymizer is applied on it.

- The *mean-distance* de-anonymizer computes the average of several distance matrices (based on different distance metrics), which results into a transformed matrix $\hat{M}$ such that $\hat{M}(i,j) = \mathsf{Mean}(M_1(i,j), M_2(i,j))$. Once this tranformed matrix $\hat{M}$ has been computed, the minimal-distance de-anonymizer is invoked on this matrix.

- The *maximal-gap de-anonymizer* also takes as input several distance matrices corresponding to different distance metrics. For each distance matrix, the minimal-distance anonymizer outputs two predictions (instead of one) for each row, which corresponds to the first and second smallest values of the distances in this row. Afterwards, the gap (*i.e*, difference) between these two values are computed. The higher the gap, the more confident we can be in the prediction made by the de-anonymizer on this particular distance matrix. Therefore, the distance metric with the highest gap among all the ones considered will be the candidate considered for the de-anonymization.

- The *simple-vote* de-anonymizer receives as input a list of candidates, which corresponds to the identities of the n minimal values of a particular row in at least two different distance matrices. The candidate that receives the highest number of votes will be the one considered for the de-anonymization. In Table **??** each column corresponds to different distance metrics while each row contains the proposed candidates at this position. For instance, in the first row Bob gets two votes against one for Alice and therefore he is considered as the most likely candidate for de-anonymization.

- Finally, the *weighted-vote* de-anonymizer, much like the simple-vote one, takes as input a list of candidates coming from different distance matrices. However, the voting method weights each possible candidate depending on the rank he has obtained for the different distance metrics. For instance, if each distance metric proposes n candidates, the first one is given a weight n, while the second receives a weight of $n-1$, and so on. For example, in Table **??**, the first column contains the weights for each candidate. The outcome of the weighted voting method is that Bob gets 8 votes, Alice 6 and Charlie 4. Therefore, Bob is considered as the "winner" (*i.e.*, most likely candidate) for the de-anonymization.

points	m_1	m_2	m_3
3	Bob	Alice	Bob
2	Charlie	Bob	Alice
1	Alice	Charlie	Charlie

Table 2: Illustration of the voting method.

6. EXPERIMENTS

We have evaluated the efficiency of the de-anonymization attack on 4 different datasets that are described in Section **??**. In this section, we report on the results of these experiments that were conducted by using the distance metrics and de-anonymizers described in the previous sections. More precisely, after validating the distance metrics on a synthetic dataset, we explain how to fine-tune the parameters of the inference attack using a first dataset (Section **??**) by measuring the accuracy of the de-anonymization attacks that rely on a single predictor (Section **??**). Then, we describe different heuristics that can be used to combine efficiently several predictors (Section **??**). These de-anonymizers based on a combination of several predictors have a high success rate that is usually better than de-anonymizer based on a single predictor as we report in Section **??**.

6.1 Datasets

Here are the four datasets used in our experiments :

- The *Geolife dataset* [**?**] has been gathered by researchers from a Microsoft Asia and consists of GPS mobility traces collected from April 2007 to October 2011, mostly in the area of the city of Shanghai. It contains the mobility traces of 178 users captured at a very high rate of 1-5 seconds.

- The *synthetic dataset* has been generated out of the Geolife dataset and we used it mainly as a sanity check for verifying if the behaviour of distance metrics and de-anonymizers match the intuition. This dataset is composed of 5 artificial individuals whose traces are all derived from the mobility traces of U_0, the first user of the Geolife dataset. U_1 is obtained by duplicating the mobility trace of U_0 (with an appropriate translation in the time domain for the second repetition of the mobility traces). The others users, U_2 U_3, U_4 and U_5 are obtained by translating the traces of U_1 in the same direction by respectively 1, 10, 100 and 1000 kilometers.

- The *Nokia dataset*[5] [**?**] is the outcome of a data collection campaign performed the city of Lausanne for 200 users started in September 2009 and ongoing. The data sampling rate is variable from user to user.

- The *Arum dataset* [**?**] is composed of the GPS mobility traces from 6 researchers sampled at a rate of 1-5 minutes in the city of Toulouse from October 2009 to January 2011.

In the following, we mainly focus on the Geolife dataset in order to analyze and understand the behaviour of the de-anonymizers and distance metrics. More precisely, for each individual of this dataset, we split his trail of mobility traces into two disjoint parts of approximately the same size. The first part forms the training set, and will be used as the adversary background knowledge, while the second part constitutes the testing set on which the de-anonymization is performed. Therefore, the objective of the adversary is to de-anonymize the individuals of the testing set by linking them to their corresponding counterparts in the training set.

6.2 Fine-tuning clustering algorithms and de-anonymizers

The states of a MMC are extracted by running a clustering algorithm on the mobility traces of an individual. Therefore, the MMC generated (and by extension the success of the de-anonymization attack) is highly dependent on the clustering algorithm used and the accuracy of this algorithm,

[5]It is worth noting that trying to reverse engineer the Nokia dataset, in order to find private information, is not allowed by the Nokia license. However, the so-called de-anonymization attack does not *per se* directly reveal any private information. However, it demonstrates that if we knew the identity of an individual user present in this dataset, we would be able to find him in another dataset, or conversely, if we would know the mobility of an identified individual, we would be able to find him in the Nokia dataset.

which may itself vary depending on the values of its parameters. The first step of our analysis consists in determining the parameters that leads to the best accuracy for the DJ-clustering algorithm.

Depending on the chosen values for the parameters, not all users of the original Geolife dataset will lead to the generation of a well-formed MMC. For instance, when the parameters of the clustering algorithm are too "conservative", some users will not have enough mobility traces to identify frequent POIs, which results in their MMC being composed of only one state. On the contrary, choosing parameters that are too "relaxed" leads to the identification of a high number of POIs thus conducting to a MMC with too many states, which is detrimental to the success of the de-anonymization attack. Thus, the main objective of the tuning phase is to find the good set of parameters for the clustering algorithm that maximize the number of users in the training set whose MMCs does not consist in only one state while keeping the average number of POIs identified per user in an acceptable range.

First, we vary the three parameters of the clustering algorithm ($MinPts$, r and the minimal number of days) and count the number of MMC generated that have more than one state. We found that these parameters are themselves correlated with the duration of the collection process and the sampling rate used. Table **??** summarizes the values of the clustering parameters used in the following experiments.

Data set	$MinPts$	r (km)	Min nb of days
Geolife	20	0.5	10
Synthetic	150	0.05	30
Nokia	10	0.05	10
Arum	40	0.05	30

Table 3: Clustering parameters validated.

We used the synthetic dataset to verify if the behavior of the distance metrics matched the expected one. Indeed, due to the way this dataset is built, for each user the training model should match the testing model. As a consequence the distance between the MMCs learnt from the training and testing set should be 0 and any de-anonymizer should have a success rate of 100%. Furthermore, the translation in space of the mobility traces of this dataset implies that the distance between the different users should approximately be around 1, 10, 100 and 1000 kilometres. These assumptions were verified in our experiments.

Once the parameters of the clustering algorithm have been tuned, we have analysed the behaviour of the de-anonymizer. In particular, the efficiency of the simple and weighted voting de-anonymizer needed to be assessed, which was the main objective of the following experiments. For the simple voting method, we first studied the influence of the number of candidates proposed by the minimal distance de-anonymizer used on each distance metric (stationary, matching, proximity and density-based). Each instance of this de-anonymizer proposes the n candidates that have the smallest distances in a row, sorted in increasing order. The simple vote de-anonymizer is applied to this list of candidates in order to output a single prediction. Figure **??** illustrates the success rate of this attack as a function of the number of candidates (more precisely as the percentage of true positives obtained over the total number of individuals) and the number of users that have well-formed MMCs (*i.e.*, MMCs that have more than one state). This experiment was conducted on the Geolife dataset with the sampling rate varying in the range $\{10s, 30s, 60s, 120s\}$.

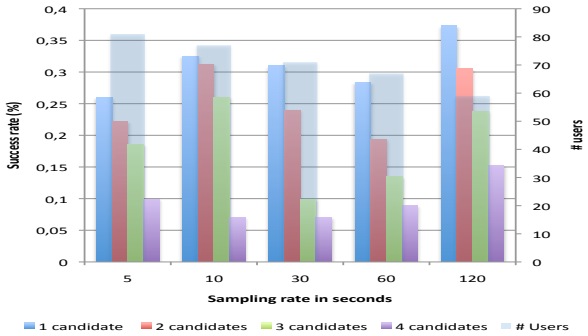

Figure 2: Success rate with the simple vote function.

From Figure **??**, we can observe that the number of users considered that have a well-formed MMC decreases as the sampling rate increases, because a low sampling rate results in fewer mobility traces to build MMCs. We can also observe that the success rate of the attack decreases as the number of candidates increases, which is not surprising as a higher number of candidates renders the task of the de-anonymizer more complex that when they are few candidates. For instance, the success rate of the attack when 4 candidates are generated by each distance metric is never more than 15%. Therefore, we can conclude that for the simple vote method considering only one candidate per distance is necessary and sufficient.

Candidate	Linear weight	Exponential weight
1	n	2^n
2	$n-1$	2^{n-1}
...	...	...
n	1	2

Table 4: The linear and exponential voting Weights.

However, in some situations it is helpful to consider more than one candidate per distance metric but their weight should be set to be different values, which is the main idea behind the weighted vote de-anonymizer. In our experiments, we have compared two different ways to weight candidates, one based on linear weights and the other on exponential ones. In a nutshell, the linear method assigns weights in a decreasing linear form and the exponential method assigns weights in a decreasing exponential form starting at 2^n, n being the number of candidates. Table **??** illustrates the assigned weights for these two methods, while Figure **??** compares the success rate of the weighted vote de-anonymizer using both weighting systems for different sampling rates. From these experiments, we can observe that the exponential system seems to be more efficient as its success rate is about 5% better than with the linear system. Moreover, this de-anonymizer seems to be robust to data sampling with different rates.

6.3 Measuring the efficiency of simple de-anonymizers

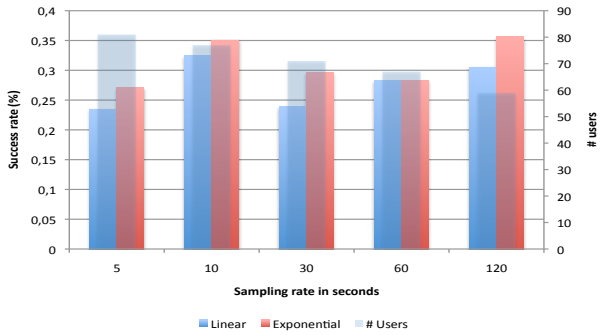

Figure 3: Success rate of the linear/exponential weighted votes.

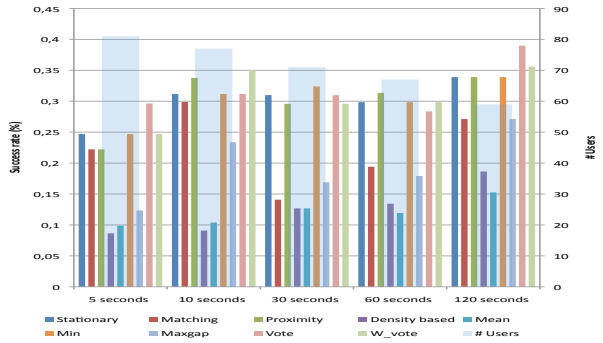

Figure 4: Success rate of the single de-anonymizers.

In this subsection, we compare the success rate of the de-anonymizers once the clustering algorithm has been tuned. To measure the success rate, we have focus on the Geolife dataset sampled at different rates. The results of these experiments are shown in Figure **??**. First, we can observe that the de-anonymizers that perform best are the minimal stationary distance, the minimal proximity distance, the minimal value, the simple vote and the weighted vote, with a success rate varying from 20% to 40%. On the contrary, both the minimal matching distance and minimal density-based distance performs very poorly with success rates around 8-15%. Nevertheless, these distance metrics through the diversity they provided can still be considered helpful as they participate in the voting process for the combined de-anonymizers. Moreover, the mean distance and the maximal gap de-anonymizers do not perform very well either, with a success rate around 10-15% for the former and 12-27% for the latter. We also observe that the maximal gap de-anonymizer performs better as the sampling rate gets higher.

Even though the success rates of the minimal stationary distance and the minimal proximity distance seems at first glance to be very similar at any sampling rate, it is important to notice that they behave very differently. Indeed, the sets of individuals that are correctly de-anonymized by these two de-anonymizers have a small intersection has illustrated by the Venn diagram presented in Figure **??**. This diagram presents in blue the set of individuals de-anonymized using the stationary minimal distance, in orange those de-

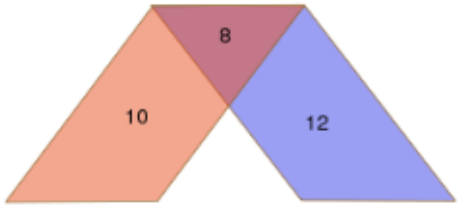

Figure 5: Venn diagram for proximity and stationary minimal distance.

anonymized using proximity minimal distance, and in red those de-anonymized by both. This observation leads us to believe that these two de-anonymizers can be used in a complementary manner, which is exactly the topic of the next subsection.

6.4 Combining de-anonymizers

In order to combine the stationary and the proximity minimal distance de-anonymizers, we designed the *statprox de-anonymizer*. This algorithm behaves exactly like the stationary minimal distance one, except when the stationary distance is above a given threshold and the proximity distance is below its maximum value (*i.e.*, 360 000 kilometers). The intuition is that if the stationary minimal distance is very small, we should use it. Otherwise, we rely on the proximity minimal distance unless it gives no conclusive result, in which case we roll back to the stationary minimal distance. The statprox de-anonymizer is presented by Algorithm **??**, while the success rate obtained by this de-anonymizer is shown in Figure **??** and compared to the other accurate de-anonymizers (namely the stationary and the proximity minimal distance and the simple and weighted vote). This results in a very high success rate (between 35 and 45%) for the statprox de-anonymizer for any sampling rate.

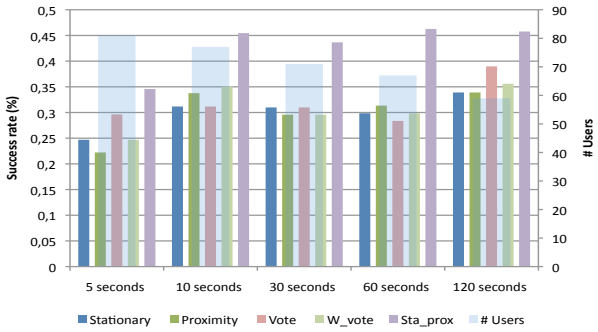

Figure 6: Success rate of the combined de-anonymizer.

At this point of the experiments, it seems important to be able to compare precisely the de-anonymizers. Indeed, the success rate of a de-anonymization attack is not the only aspect that should be considered. For instance, for an adversary a possible strategy is to focus on weak individuals that offer a high probability of success for the attack rather than being able to de-anonymize the entire dataset. Measuring the probability of success of the inference attack for a given individual is similar to have some kind of confidence measure

Algorithm 6 Stationary-Proximity decision algorithm.

Require: $threshold_{stat}$, $threshold_{prox}$, $result_{stat}$, $result_{prox}$

```
if result_prox < threshold_stat then
  return result_prox
else
  if result_stat < threshold_stat then
    return result_stat
  else
    if result_prox < 360000 then
      return result_prox
    else
      return result_stat
    end if
  end if
end if
```

for a given de-anonymization candidate. Deriving this confidence measure is quite intuitive for our de-anonymizers. Indeed, for the minimal distance ones, the smaller is the distance, the higher the confidence, while for the vote based ones, the higher the voting score is, the higher the confidence.

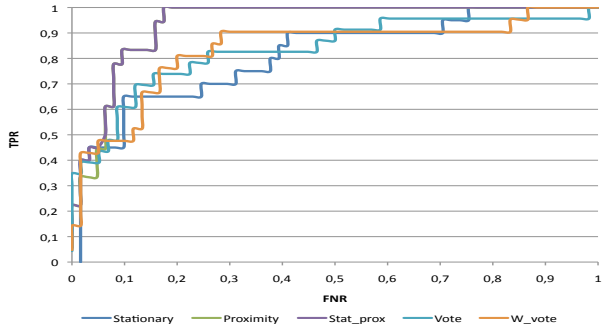

Figure 7: ROC curve for Geolife dataset.

In order to compare the performance of the de-anonymizers, we used a graph inspired from the concept of *Receiver Operating Characteristic* (ROC) curves. A ROC curve is a graphical plot representing the sensitivity (*i.e.* true positive rate versus false positive rate), for a classifier system as the value of its discrimination threshold changes. The idea is that, between two de-anonymizers having the same success rate, we should favor the one succeeding more quickly (when the candidates are sorted using the confidence metric). Henceforth, the following so-called ROC curve (Figure **??**) shows the true positives rate (TPR) versus the false positives rate (FPR) for the best performing de-anonymizers, with the candidates being sorted by ascending distance or descending number of votes. This ROC curve further confirms that the statprox de-anonymizer seems to be the best alternative among all the different de-anonymizers that we have developed.

6.5 De-anonymizing other datasets

We have seen that the de-anonymizers developed work fairly well for the Geolife dataset as the achieved success rate is between 35% and 45% for the statprox de-anonymizer and between 29% and 39% for the simple vote. In this subsection, we have tested how the algorithms developed and trained with a specific dataset (*i.e.*, the Geolife dataset), are still very accurate with other datasets such as the Nokia and Arum ones.

First, we test the algorithms on the Nokia dataset. This dataset has 195 users, among which 157 users can be used to generate a correct MMC using the parameters described previously, which varies from 28% to 42% (statprox).

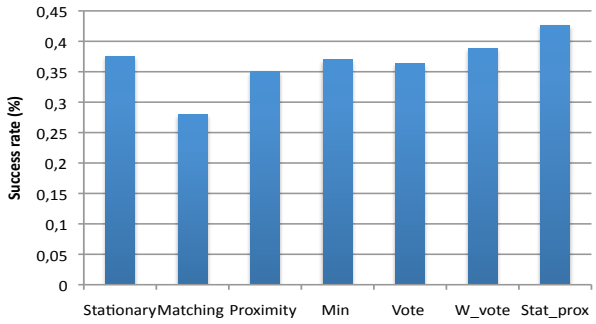

Figure 8: Success rate of the de-anonymizers with the Nokia dataset.

These results were confirmed using the ARUM dataset in which the statprox de-anonymizer obtains a success rate of 80% (*i.e.*, is able to "re-identify" 4 individuals over 5). An important characteristic of the ARUM dataset is that the individuals are all from the same institute and all leave in the same city. Henceforth, their mobility models share some important characteristics. For instance, their working place is the same and they all live nearby. However, since their usual mobility lies within a smaller area, the parameters we used for building the MMCs are more aggressive.

6.6 Discussion.

In this section, we have presented various experiments on de-anonymization attacks that lead to the definition of an heterogeneous de-anonymizer, called *statprox*, that obtains a success rate of de-anonymization between 42% and 80% on different datasets. While this performance may, at first glance, seem to be poorer than the one achieved by the predictors of Ma *et al.* in [**?**], which goes up to 70-80%, we believe that these results are not necessarily comparable because we clearly differentiate between the training set and the testing set, while they perform the learning and the testing on the same dataset, thus inducing a strong experimental bias. Indeed, first, our mobility models are built out of the training set, which is disjoint from the test set, whereas one of the adversary model of Ma *et al.* (*i.e.*, called *A1*) directly extracts timestamped data from the test set. Moreover, in our case the training data is temporally separated from the test data (*i.e.*, the training and the test have been recorded at different non-overlapping periods of time) because the whole dataset has been split into two temporally disjoint parts, whereas the second adversary model of Ma *et al.* (*i.e.*, called *A2*) picks the information it uses to de-anonymize within the same period as the test data is recorded. Therefore, our approach is quite different from them as our attack consists first in collecting mobility data from an individual, before *later*, trying to identify this individual in a so-called anonymized dataset, while their attack

aims at gathering location data at the same time at which the de-anonymization attack occurs. In addition, one important parameter of their attack is the number of timestamped location data collected, which can be compared to the number of states we have in our mobility model. On average and depending on the dataset considered, we have between 4 and 8 states per MMC, which correspond to a compact representation of the mobility behavior of an individual. When restricted to such limited of information in terms of number of timestamped location data, the attacks proposed by Ma *et al.* do not perform well, with for instance a de-anonymization rate between 10% and 40% for the adversary *A1*.

7. CONCLUSION

In this paper, we have demonstrated that geolocated datasets gathering the movements of individuals are particularly vulnerable to a form of inference attack called the de-anonymization attack, and this even if the mobility traces have been anonymized prior to release. More precisely, we have shown that the de-anonymization attack can re-identify with a high success rate the individuals whose movements are contained in an anonymous dataset provided that the adversary can used as background information some mobility traces of the same individuals that he has been able to observe during the training phase. Out of these traces, the adversary can build a MMC that models in a compact and precise way the mobility behavior of an individual. We designed novel distance metrics that measure the similarity between two MMCs and we described how these metrics can be combined to build different types of de-anonymizers. The effectiveness of the de-anonymizers were first verified on a small synthetic dataset before being applied on a larger scale on real datasets. The de-anonymization attack is very accurate with a success rate of up to 45% on real and large datasets and this even if the mobility traces are sanitized byy sampling them with a low frequency (for instance every 2 minutes instead of every 10 seconds).

In the future, we will extend the current work by following several avenues of research. For instance, we will study other clustering algorithms for the extraction of the POIs, which impact the creation of the states of the MMC and therefore the MMC itself. In particular, it is possible that two different clustering algorithms apply on the same trail of mobility traces can result in two very distinct MMCs that might be quite far from each other. Therefore, our main goal will be to discover among different clustering algorithms, the one that best fits our needs while being also stable with respect to small changes in the inputs (*e.g.*, small spatial and temporal perturbation). On the opposite direction, we will also explore how more complex geo-sanitization mechanisms, such as spatial cloaking techniques or mix zones, can help to reduce the success rate of the attack. Finally, we will also try to infer social relationships between individuals from the mobility traces and design variant of the de-anonymization attack exploiting this knowledge.

8. REFERENCES

[1] D. Ashbrook and T. Starner. Learning significant locations and predicting user movement with gps. In *Proceedings of the 6th IEEE International Symposium on Wearable Computers*, volume 7, pages 275–286, Sardina, Italy, February 2003.

[2] Y. De Mulder, G. Danezis, L. Batina, and B. Preneel. Identification via location-profiling in GSM networks. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, volume WPES '08, pages 23–32, Alexandria, VA, USA, October 2008.

[3] N. Eagle and A. (Sandy) Pentland. Reality mining: sensing complex social systems. In *Personal and Ubiquitous Computing*, volume 10, pages 255–268, London, UK, March 2006.

[4] S. Gambs, M.-O. Killijian, and M. Núñez del Prado Cortez. Show me how you move and I will tell you who you are. In *Transactions on Data Privacy*, volume 2, pages 103–126, Catalonia, Spain, August 2011.

[5] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. In *Nature*, volume 453, pages 779–782, New Orleans, LA, USA, June 2008.

[6] L. Jedrzejczyk, B. Price, A. Bandara, and B. Nuseibeh. I know what you did last summer: risks of location data leakage in mobile and social computing. In *Department of Computing Faculty of Mathematics, Computing and Technology The Open University*, pages 1744–1986, Milton Keynes, UK, November 2008.

[7] M. Killijian, M. Roy, and G. Trédan. Beyond San Francisco cabs: building a *-lity mining dataset. In *Workshop on the Analysis of Mobile Phone Networks (NetMob)*, volume Sattelite of NetSci, pages 75–78, Cambridge, MA, USA, 2010.

[8] N. Kiukkonen. Technical report: Data collection campaign. Technical report, Nokia Research Center, Lausanne, Switzerland, December 2009.

[9] J. Krumm. Inference attacks on location tracks. In *Pervasive Computing*, volume 4480, pages 127–143, Toronto, Canada, June 2007.

[10] N. S.-D. M. Piorkowski and M. Grossglauser. Crawdad data set epfl/mobility.

[11] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao. Privacy vulnerability of published anonymous mobility traces. In *Proceedings of the 16th annual international conference on Mobile computing and networking*, MobiCom '10, pages 185–196, New York, NY, USA, 2010.

[12] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125, Washington, DC, USA, 2008.

[13] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. In *Science*, volume 327, pages 1018–1021, New Orleans, LA, USA, February 2010.

[14] H. Taha and V. Pozo. *Investigación de operaciones*. Pearson Educación, Naucalpan de Juarez, Edo de México, 7 edition, October 2004.

[15] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. SeMiTri: a framework for semantic annotation of heterogeneous trajectories. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 259–270, New York, NY, USA, 2011.

[16] H. Zang and J. C. Bolot. Anonymization of location data does not work: A large-scale measurement study. In *Proc. of ACM Mobicom*, pages 145–156, Las Vegas, NV, USA, 2011.

[17] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma. Understanding mobility based on gps data. In *In Proceedings of ACM conference on Ubiquitous Computing*, volume ACM Press, pages 312–321, Seoul, Korea, 2008.

[18] Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. In *IEEE Data Engineering Bulletin*, volume 33, pages 32–40, Beijing, P.R. China, 2010.