Exploring Spatial Transformation-Based Privacy in a Small Town

Aidan Jacobs
Computer Science & Information Technology
San Juan College
Farmington, NM 87402. USA
Email: arjacobs1@hotmail.com

Subhasish Mazumdar
Computer Science & Engineering
New Mexico Institute of Mining and Technology
Socorro, NM 87801. USA
Email: Subhasish.Mazumdar@nmt.edu

***Abstract*—As mobile devices become increasingly prevalent in society, the expected utility of such devices rises; arguably, the most impact comes from location-based services as they provide tremendous benefits to mobile users. These users also value privacy, i.e., keeping their locations and search queries private, but that is not easy to achieve. It has been previously proposed that user location privacy can be secured through the use of space filling curves due to their ability to preserve spatial proximity while hiding the actual physical locations. With a space filling curve, such as the Hilbert curve, an application that provides location-based services can allow the user to take advantage of those services without transmitting a physical location. Earlier research has uncovered vulnerabilities of such systems and proposed remedies. But those countermeasures were clearly aimed at reasonably large metropolitan areas. It was not clear if they were appropriate for small towns, which display sparsity of Points of Interest (POIs) and limited diversity in their categories. This paper studies the issue focusing on a small university town.**

***Index Terms*—Mobile environments; Location-dependent and sensitive; Privacy; Query Processing.**

I. Introduction

Users are interested in location-based queries like “find me the nearest C” or “find me k nearest Cs,” where C is a service category like *restaurant* or *gas station*. However, they face the risk of compromising their privacy; consequently, researchers have suggested spatial transformation to address such concerns. The Hilbert curve-based approach maps every geographic coordinate (x, y) into a number $h(x, y)$ through a Hilbert function h that has useful properties: it fills the grid in such a way that consecutive mapped numbers are physically contiguous (though not necessarily the other way around). Accordingly, the *Location Based Server (LBS)* is given values of $h(x, y)$ instead of (x, y), thus encoding both the locations of *Points of Interest* (POIs) and those of users; further, instead of the categories of interest, it gets encrypted descriptions. Thus, the LBS is able to answer queries without being aware of either user locations or the categories of their interest.

Earlier works have explored strategies that a rogue LBS can adopt in order to defeat this method. In [1], semantic factors such as the distribution of POI categories and variations in POI density were considered and countermeasures offered. Further enhancements of the method, in the form of rotation and transposition, were offered in [2]. However, this approach is especially challenging when the grid represents a small town that has only a modest number of POIs — that too in a cluster or two — and limited diversity among the categories. In this paper, we explore that issue by choosing Socorro, New Mexico, our university town, with a population less than 9,000, on which to test the methods.

The paper is structured as follows. In the second section, we outline the basic strategy behind the spatial transform method. In the third, we present related work. The fourth covers the data describing Socorro as well as semantic attacks on that data. The fifth and sixth sections outline our proposed countermeasures based on adding fake POIs and a feature of the L1-variant respectively. We offer concluding remarks in the seventh section.

II. The setup

Suppose we have a $2^N \times 2^N$ grid (containing 2^{2N} cells) corresponding to geographic coordinates (x, y) where x and y are integers in $0 \cdot\cdot (2^N - 1)$. A Hilbert curve H of order N gives a bijective function h that maps each (x, y) into an integer in $0 \cdot\cdot (2^{2N} - 1)$, which is interpreted as the Hilbert cell number. Figure 1 shows an example with $N = 3$ (i.e., an 8x8 grid); it shows the cell numbers $h(x, y)$ inside each cell. The sequence $0 \cdot \cdot 63$ defines the *Hilbert curve* that fills the grid passing through each cell exactly once. We view it as a matrix where the cell at the bottom row and leftmost column corresponds to $H[0, 0]$ representing the geographic coordinate $(0, 0)$. Rotated representations of the curve are useful too; Figures 2, 3, and 4 depict rotations by 90°, 180°, and 270°, respectively.

This function h is contiguity-preserving, i.e., two cells numbered i and $(i + 1)$ in Hilbert space must represent geographical coordinates that are contiguous. However, h may map contiguous coordinates in 2D-space into Hilbert numbers that are *not* even close (e.g., numerically distant cells 5 and 58 in Figure 1 represent contiguous coordinates).

A. Utilizing the Hilbert Curve for Location Privacy

A Hilbert curve is generated by a Trusted Server (TS) after deciding the curve’s parameters. They are: (1) the *order* of the curve N; (2) the (physical location of the) point of

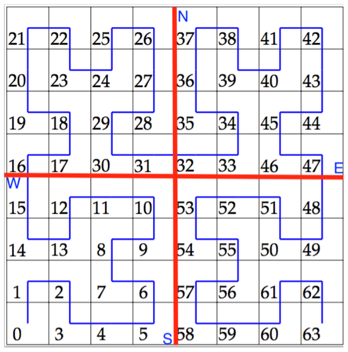

Figure 1. Normal Hilbert Curve for $N = 3$. The cell in the bottom row and leftmost column corresponds to geographical $(0, 0)$ and matrix element [0,0].

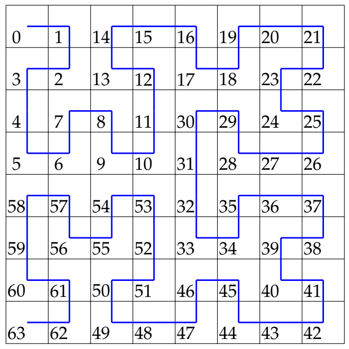

Figure 2. Hilbert Curve for $N = 3$ rotated clockwise by 90 degrees.

origin X_0, Y_0; (3) the *orientation* Θ (*normal* as in Figure 1 or *transposed* (the matrix transpose of Figure 1, not shown); and (4) a *scaling factor* Γ that captures the number of meters that each unit cell represents (in both figures, Γ is the distance in meters covered by the entire grid in either the X- or Y-direction divided by 8). Using Γ and the origin, any geographic location (x_0, y_0) (which could be latitude and longitude), can be converted into a grid cell or matrix element $H[(x_0^*, y_0^*)]$, where $x_0^*, y_0^* \in 0 \cdot \cdot (2^{2N} - 1)$. Thus, the transformation parameters (unknown to the LBS) are $[X_0, Y_0, \theta, N, \Gamma]$.

First, the TS sends a table of POIs with encrypted categories (see Table I). Later, the LBS gets a query containing the Hilbert cell number of the mobile user along with an encrypted (sub)category. The LBS searches for the numerically closest Hilbert cell number containing POIs of that (sub)category and returns it to the user.

III. Related Work

Khoshgozaran et al. [3] first suggested the use of a Hilbert curve for location-based services. Abel et al. [4] compared Hilbert curves with four spatial transformation orderings — row, row prime, Morton and Gray code — and found that Hilbert curves are weaker than Morton in some aspects but superior for preserving contiguity. Moon et al. [5] showed the effect of rotation of the Hilbert curve on clustering. The effects of shifts of the Hilbert curve on the loss of proximity in the Hilbert space were shown in [6] and [7].

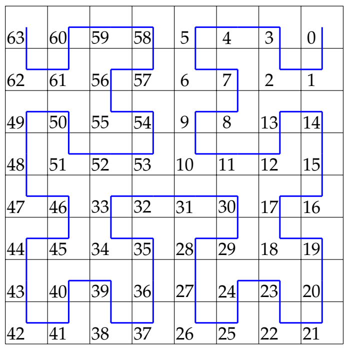

Figure 3. Hilbert Curve for $N = 3$ rotated clockwise by 180 degrees.

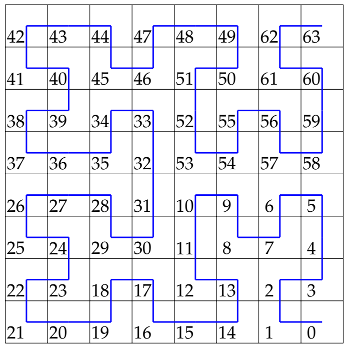

Figure 4. Hilbert Curve for $N = 3$ rotated clockwise by 270 degrees.

IV. Small town: Socorro, NM

We gathered 147 POIs from Socorro, New Mexico, USA, and stored them in a table. Each POI entry consisted of its latitude-longitude pair, category, and subcategories, if applicable. We used categories and subcategories used by *Yelp* and found 19 categories and 3 levels of subcategories. Table II shows a fragment of our table. Figure 5 shows the category hierarchy as graphs: each node contains the number of POIs that fall under that and descendant subcategories. Next, we mapped them with a Hilbert curve of order 4; we found a cluster of POIs with large sparse regions (see Figure 6).

TABLE I
Example of a Table Sent From the TS to the LBS

Hilbert Cell	(encrypted) POI description	(encrypted) Category	(encrypted) Subcategory
43	1547DA276CCDA	9A4027D	0032
...	...	...	...
15	07BB583A9FF46	1C011DD	0120
16	9577CC2D55B2F	9A4027D	0122

TABLE II
A Fragment of the Data for Socorro

Name	category	subcat. 1	subcat. 2	subcat. 3
P.B.S.	beauty / spa			
Nusenda	financial serv.	banking		
Solaro	home services	real estate	solar inst.	
YMG	restaurant	american	steak	seafood

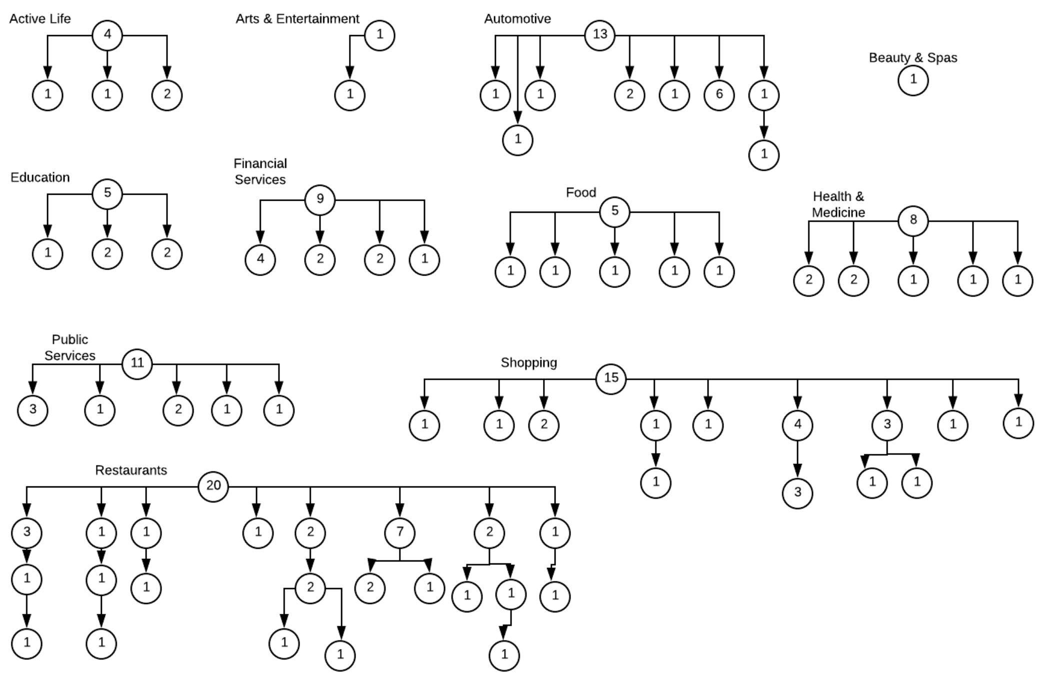

Figure 5. Some category trees for Socorro, NM. Each node represents a sub/category and shows the number of POIs corresponding to the subtree rooted at that node.

A. Attacking the Socorro data

Playing the role of a rogue LBS, we sorted the encoded data by category, sub-category, etc., and tallied the number of instances of each. Next, we associated each POI with a tuple that represented the categories and subcategories which it belonged to. For example, a POI with the tuple (6, 4, 1) would share its category with 5 other POIs, its sub-category with 3 other POIs, and be the only POI in its sub-subcategory.

The categories *Arts & Entertainment* and *Health & Beauty* each had one POI within them. The *Arts & Entertainment* POI was located on the outskirts of town while the *Health & Beauty* POI was located in the center. By counting the number of surrounding POIs, we could easily identify which entry in the table of POIs belonged to *Arts & Entertainment* and which entry belonged to *Health & Beauty*.

Pets was the only category that contained three POIs — two veterinary clinics and an animal shelter. The two clinics were close together while the animal shelter was on the opposite side of town. By comparing cell values within this category, we could easily identify which POI was the animal shelter.

Using these methods, we could identify 21 of the 147 POIs in Socorro. From just two of those identified POIs which fell in different cells, we could derive the scaling factor Γ (following the scheme outlined in [1]).

V. COUNTERMEASURES

A. Replication and Rotation Method

If the encoded map contains a significant amount of empty cells (i.e., cells containing no POIs), the LBS can use this feature to make educated guesses about the alignment of POIs. For example, the encoded map for Socorro shows that almost all the POIs fall on the left half (Figure 6).

One way to mitigate this issue is to generate fake POIs and place them in the empty cells. The fake POIs must be distributed in a manner such that the LBS cannot easily discard them. In other words, they must mimic the distribution of real POIs. Furthermore, since users cannot send queries from impossible locations, the LBS may identify some of these fakes if no user ever queries from those locations. (It is true that not every empty space is an impossible location.) So, we require that the TS should generate fake users to periodically query the fake POIs.

We propose a *Replication and Rotation Method*, whose goal is to fill empty space on the encoded map as well as to generate fake user queries that are indistinguishable from real user queries. The method is the following.

- The TS replicates the set of real POIs into four sets — the original set S, S encoded with 90° rotation, S with 180° rotation, and S with 270° rotation — and superimposes them onto the same grid. For example, suppose $N = 3$ (8x8 grid) and there are just 2 POIs in (2,3) and (5,0). From Figures 1, 2, 3, and 4, they are 11, 59 (in the 0° curve); 54, 44 (90°); 33, 25 (180°); and 28, 14 (270°). So, the list of POIs the LBS gets is [11, 14, 25, 28, 33, 44, 54, 59]. Furthermore, the categories in the 0° set are encrypted once; those in the 90° set are encrypted twice; and those in the 180° and 270° thrice and four times respectively. The LBS gets a single table containing all four sets.

- The LBS gets four queries for each user query; they contain the user's location encoded at 0°, 90°, 180°, and 270° rotations. Thus, if the user is at (4,1), the LBS gets four queries from 57, 46, 27, and 12. In addition, the POI category desired by the user also goes through this process: encrypted once for the first query (0°), twice, thrice, and four times for the other three (90°, 180°, and 270° rotations respectively).
- Since the LBS receives these four queries (not simultaneously), they appear to be from different locations.
- The user's computational module only accepts the response to the first. It rejects the other three, which are detected when (a single) decryption results in gibberish. The delays and arrival order of the responses are inconsequential.

We applied this system to our Socorro data set. The improvement in coverage of the curve can be seen by comparing the distribution of colored cells (presence of POIs) in Figure 7 against that in Figure 6.

To estimate the expected improvement in coverage, let p be the fraction of the grid that is filled before applying this method and P be the fraction of the grid that is filled after.

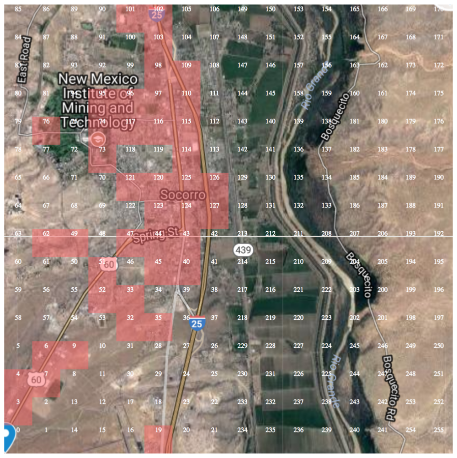

Figure 6. Socorro POIs encoded at order 4.

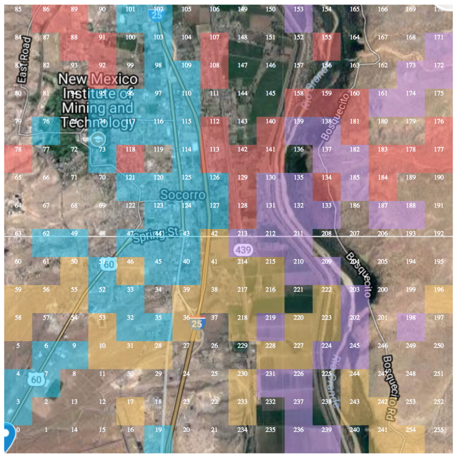

Figure 7. Replication and rotation method applied to Socorro POIs.

Given p, the value of P is expected to be

$$P = p + p(1-p) + p(1-p)^2 + p(1-p)^3 \quad (1)$$
$$= 4p - 6p^2 + 4p^3 - p^4 \quad (2)$$

In (1), the first term represents the original data in which a fraction p of cells contain POIs. The next three terms reflect the three steps of *Replication and Rotation* where the fractions of still-empty cells are reduced by p. Figure 8 plots this improved coverage P against the initial coverage p.

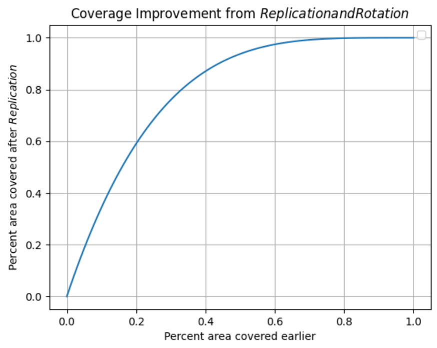

Figure 8. Improvement in coverage from *Replication and Rotation*.

B. Balancing the Category Distribution

We found that this system is still susceptible to semantics-based attacks arising from the asymmetry in the category / subcategory hierarchy. Also, if there is only one POI of a particular category and all other categories have more than one POI, then the LBS can find out that four is the minimum number of POIs of a particular (encrypted) category and guess a four-fold replication scheme.

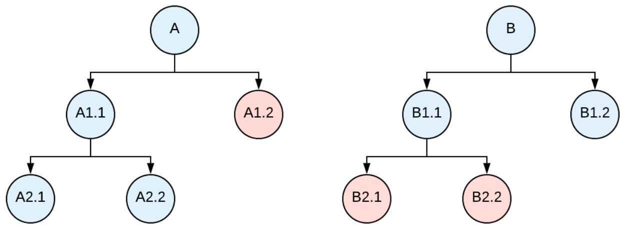

Figure 9. An example of balancing the category trees. Blue cells represent original categories with real POIs. Since trees A and B are different, fake categories (red cells A1.2, B2.1, and B2.2) with fake POIs are added.

The solution for the problem of variation in category hierarchy is to add fake sub-categories so that every category has the same number of sub-categories, sub-sub-categories, etc., thereby making the trees similar. For example, in Figure 9, the original trees for A and B consisting of the blue nodes are clearly different; we compensate for the difference by adding the red nodes.

Later, fake POIs would be added for those fake categories. Users should be presented with the appropriate lowest-level subcategories to choose from based on their initial query. Thus,

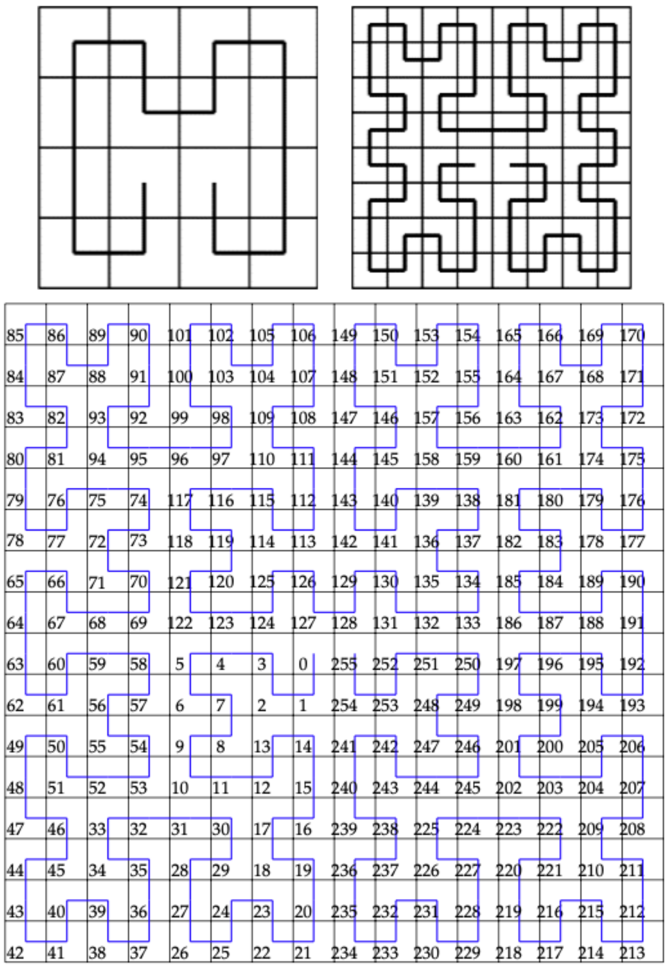

Figure 10. Orders 2, 3, and 4 of the $L1$ variant of the Hilbert curve.

a fake POI will never be returned to a real user. As mentioned earlier, the TS must generate fake user queries for fake POIs.

Balancing the category tree for the Socorro data set (Figure 6) required over twice as many fake POIs as real POIs. This is feasible for our small town but may not be useful for large cities. In summary, balancing the category tree is possible and crucial to stop the LBS from identifying POIs based on the category distribution.

VI. The $L1$ Variant

We also considered the scheme of *offsets*, in which the TS applies an offset to all encoded locations of both POIs and users to confuse a rogue LBS, i.e., it sends Hilbert cell $h(x,y)+k$ instead of $h(x,y)$, where k, a constant, is the offset. Such offsets and wraparounds have been discussed in [2].

We assume that the LBS *knows* that an offset has been applied and has identified Hilbert cells c_1 and c_2, i.e., found the physical locations of the two corresponding POIs, and therefore knows their spatial (geographical) relationship (angle of the vector joining them). For the degree of uncertainty of the LBS, we choose the number of Hilbert cell pairs (c_1', c_2') with a similar spatial relationship that could have yielded (c_1, c_2) through the offset scheme (considering rotation too). If the number is low, then the LBS can guess the amount of offset that was applied and which of the curve rotations was selected.

Taking a curve of order 4, we considered every possible gap g and every pair of cells (c_1, c_2), where $g = |c_1 - c_2|$. We define $n_p(g)$ as the number of such *possible* pairs for a given gap g. For example, allowing only $c_2 > c_1$, $n_p(9) = 247$ but $n_p(254) = 2$, i.e., there are many more pairs for a small gap than for a large one. The total number $\sum_{g=1}^{255} n_p(g)$ = 32,640.

Next, we considered every possible offset and rotation, while ruling out disruptive offsets which occur when a cell is moved off the grid; when that happens, wraparound is the only practical option, i.e., $h(x,y) + k$ gets replaced by $(h(x,y) + k) \bmod 2^{2N}$. Since cell $2^{2N} - 1$ is geographically distant from cell 0, these cause extreme discontinuities, and for a pair of cells, a sudden change in the angle of the vector.

We define (c_1', c_2') a *feasible* pair for (c_1, c_2) if (i) (c_1', c_2') can be transformed into (c_1, c_2) through offset and rotation; and (ii) the angles of the two vectors (one from c_1 to c_2 and the other from c_1' to c_2') are acceptably similar, i.e., are less than or equal to an upper bound α.

Even α = 90° can imply similarity. For example, in Figure 1, suppose the LBS has identified POIs in 8 and 9. It must consider the possibility that $(8, 9)$ was transformed from $(2, 3)$ via an offset of 6 (with no rotation) even though the vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$, joining the centers of $(8, 9)$ and $(2, 3)$ respectively, are at 90° to each other. This is because if the POIs are *physically* in the bottom left corner of cell 2 and top right corner of cell 3, then the vector joining *them* would be almost indistinguishable from $\overrightarrow{v_1}$. We choose α = 45° for our analysis. Our results do not change significantly when α is increased to 90°.

We define $n_f(g)$, the number of feasible pairs for a gap g, as the count of feasible pairs for all possible identified pairs having a gap g. This number is a measure of total uncertainty regarding offset and rotations that the LBS faces (in its attempt to break the offset scheme) for a given gap.

Finally, to get the uncertainty per cell pair for a given gap, we averaged over the possible pairs for each gap. We plotted $n_f(g)/n_p(g)$, the average amount of uncertainty for each gap g; we found that it declined steadily with increasing gap g. Low uncertainty, even if it is only for larger gaps, is a weakness of the offset scheme that the LBS could well exploit.

To address this weakness, we repeated the earlier computation on the $L1$-variant of the traditional Hilbert curve [8], which begins and ends in adjacent cells in the middle of the grid, i.e., cell $(2^{2N} - 1)$ is next to cell 0 (Figure 10). Unlike the traditional curve, there are no sudden disruptions, i.e., a cell does not move off the grid when $(h(x,y)+k) > 2^{2N}$. So, we allowed all offsets $k \in [0 \cdots 2^{2N}]$ (the total number of cells is 2^{2N}) while transforming $h(x,y)$ to $(h(x,y)+k) \bmod 2^{2N}$. We found that the $L1$-variant provides a steady amount of uncertainty even for large cell gaps, thus remedying the weakness of the normal curve. The results are shown in Figure 11.

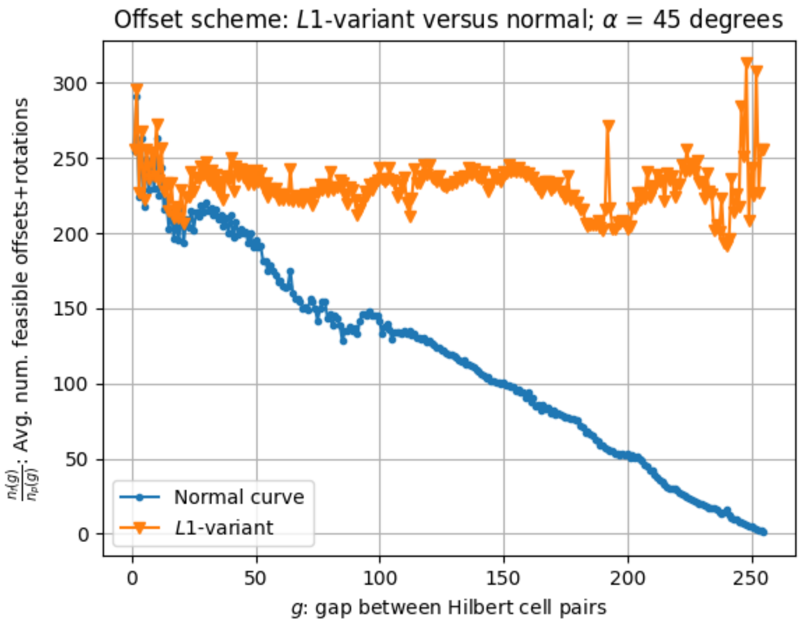

Figure 11. $\frac{n_f(g)}{n_p(g)}$, a measure of strength of the offset scheme, is plotted against g, the gap (i.e., difference) between the Hilbert numbers of a pair of identified POIs. The numerator is the number of *possible* cell pairs with a given gap g; the denominator is the number of *feasible* pairs, i.e, pairs that could have been transformed by offset and rotation into the identified pair such that the angle between the two pairs is less than or equal to α. The ratio gives the average number of *feasible* POI pairs per *possible* cell pair for a given g. Unlike the normal curve, the $L1$-variant shows steady resilience against large gaps.

VII. Conclusion

Spatial transformation via Hilbert curves has been shown to be useful in reclaiming privacy for mobile users when their location-based queries are answered. However, the method shows vulnerabilities when applied to small towns. In this paper, we have analyzed one such town — a university town in New Mexico, USA — and found that a rogue LBS can attack the privacy protection quite effectively, as predicted by earlier researchers, by exploiting the sparsity of POIs and the imbalance in category trees.

We have proposed and tested effective countermeasures, all of which either limit or eliminate potential weaknesses. The *Replication and Rotation* method limits attacks based on sparsity; balancing category distribution effectively counters semantics-based attacks. The $L1$ variant protects the *offset* scheme against compromise of numerically distant cells.

One limitation of our approach is that both the *Replication and Rotation* method and the normalization of category distribution require the additional overhead of posting fake queries. However, the $L1$-method requires little computational overhead because it neither needs fake POIs nor periodic fake queries.

In the future, we hope to better quantify the improvement to location based privacy provided by this method. We also hope to quantify the fraction of POIs required for a general data set, rather than just for Socorro. A full-fledged implementation would allow us to get a quantitative estimate of the response times and unforeseen computational challenges.

Acknowledgment

This work was supported by the National Science Foundation under grants CNS-1757945 and CNS-2150145. We gratefully acknowledge the contribution of Eric Majidian who participated in the research but could not be involved in the writing of this paper owing to contractual obligations with his current employer.

References

[1] A. Paturi and S. Mazumdar, “Can spatial transformation-based privacy preservation compromise location privacy?” in *Proc. 15th Intl. Conf on Trust, Privacy and Security in Digital Business (TrustBus ’18), part of DEXA 2018*, 2018, pp. 69–84, ISBN:978-3-319-98384-4.

[2] A. Paturi and S. Mazumdar, “Exploring origin and rotation parameters while using Hilbert curves in mobile environments,” in *Proc. 8th Intl. Conf on Mobile Services, Resources, and Users (IARIA Mobility 2018), part of DataSys 2018*, 2018, pp. 8–13, ISBN: 978-1-61208-656-9.

[3] A. Khoshgozaran and C. Shahabi, “Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy,” in *Proc. 10th International Conference on Advances in Spatial and Temporal Databases*, 2007, pp. 239–257, ISBN: 978-3-540-73539-7.

[4] D. J. Abel and D. M. Mark, “A comparative analysis of some two-dimensional orderings,” *International Journal of Geographical Information Systems*, vol. 4, no. 1, pp. 21–31, 1990, ISSN: 1365-8816.

[5] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz, “Analysis of the clustering properties of the Hilbert space-filling curve,” *IEEE Trans. Knowledge & Data Engineering*, vol. 13, no. 1, pp. 124–141, Jan 2001, ISSN: 1041-4347.

[6] J. Dai, “Efficient range query using multiple Hilbert curves,” in *Current Trends and Challenges in RFID*, C. Turcu, Ed. InTech, 2011, pp. 375–390, ISBN: 978-953-307-356-9.

[7] S. Liao, M. A. Lopez, and S. T. Leutenegger, “High dimensional similarity search with space filling curves,” in *Proceedings 17th International Conference on Data Engineering*, 2001, pp. 615–622, ISSN: 1063-6382.

[8] X. Liu, “Four alternative patterns of the Hilbert curve,” *Applied Mathematics and Computation*, vol. 147, no. 3, pp. 741–752, 2004, ISSN: 0096-3003.